Universitat de Lleida

# TREBALL FINAL DE GRAU

ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

**Estudiant:** **Carlo Sergio Cacho Pacheco**

**Titulació:** Grau en Enginyeria Informàtica

**Títol de Treball Final de Grau:** Modulo del Impuesto Sobre el Beneficio

**Directora:** MONTSERRAT SENDIN VELOSO

**Codirector:** ALFONSO GARRAY PLANES

Presentació

Mes: Juny

Any: 2019

# AGRADECIMIENTOS

Agradecimientos a mi familia a mis padres y hermanos que siempre estuvieron y estarán a mi lado, que son quienes me apoyaron incondicionalmente a lo largo de mi profesión brindándome consejos, y que siempre están junto a mi tanto en los buenos momentos y como también en los malos.

Especialmente agradezco a mi madre y a mi padre que nunca se han rendido y siempre ha creído en mí, siempre dándome esos empujones de seguir adelante para poder conseguirlo.

# RESUMEN

El presente proyecto busca el desarrollo del módulo para la Gestión del Impuesto Fiscal Sobre Beneficio que se integrara a la aplicación de SIGEFI. Este proyecto nace del estudio de las necesidades de los clientes actuales que tienen adquirida la aplicación de SIGEFI, los cuales tenían carencias para la gestión del manejo de los impuestos sobre el beneficio, estos clientes realizaban esta gestión por otros medios sean de forma manual usando tablas Excel u otro software de terceros que no acaparaban todas sus necesidades y que les creaba más trabajo al no estar todo unido en un mismo sistema.

El equipo de SIGEFI con aproximadamente 14 años de experiencia en el ámbito fiscal, se le encargo el desarrollo de dicho modulo y con ello integrarlo al software unificado de gestión Fiscal (SIGEFI).

SIGEFI es un software que se encuentra en constante cambio, ya que las leyes fiscales suelen cambiar de un año a otro y se debe de modificar el software cada vez que lo requieran. Además de ello se van implementando con los años nuevas mejoras en el software como el desarrollo de nuevos módulos según se requiera o según se detecte una necesidad en el mercado.

Los desarrollos generados por necesidad del cliente o por el análisis del equipo funcional de SIGEFI son beneficiosos para todos los clientes ya que estas nuevas características son aplicables a todos los clientes, cada cliente puede decidir que modulo se adapta más a su necesidad.

# Contenido

# INDICE DE FIGURAS

# GLOSARIO DE TERMINOS

| | |
|---|---|
| **CAF** | Composite Application Framework |
| **DC** | Development Component |
| **SC** | Software Component |
| **EJB** | Enterprise Java Bean |
| **EP** | Enterprise Portal |
| **EPCF** | Enterprise Portal Client Framework |
| **J2EE** | Java 2 Enterprise Edition |
| **JSP** | Java Server Pages |
| **KM** | Knowledge Management |
| **MVC** | Model View Controller |
| **NW** | NetWeaver |
| **NWDI** | NetWeaver Development Infrastructure |
| **NDS** | NetWeaver Developer Studio |
| **PDK** | Portal Development Kit |
| **UI** | User Interface (Interfaz de usuario) |
| **WD** | Web Dynpro |
| **WDJ** | Web Dynpro for Java |
| **WS** | Web Service |
| **SAAS** | Software as a Service |

# 1. INTRODUCCION

## 1.1 MOTIVACION

Al realizar las prácticas en la empresa de Indra y luego de la incorporación de mi persona a la plantilla de Indra es donde nace la oportunidad de poder desarrollar mi trabajo de final de grado en Indra, esta necesidad fue informada por un cliente el cual cuenta con el Sistema de Gestión Fiscal (SIGEFI) , que tenía las necesidad que la aplicación actualmente de SIGEFI añada un nuevo módulo para la Gestión del Impuesto sobre Beneficio, por lo cual solicite formar parte del desarrollo para adquirir experiencia y aprovechar para entender todo el sistema de SIGEFI y formar parte del desarrollo de un módulo al completo.

## 1.2 OBJETIVOS

El objetivo es el desarrollo del módulo de la Gestión del Impuesto sobre Beneficio. La aplicación se encargará de brindarles una correcta gestión de los datos que ingrese los usuarios en el sistema para calcular que cantidad exacta se debe pagar al estado por los beneficios generados de la empresa. Por otro lado, los clientes se verán beneficiados ya que podrán tener un módulo nuevo en la aplicación de SIGEFI donde podrán gestionar de manera unificada toda la gestión de la empresa, brindándoles más control de sus datos y minimizando los tiempos de presentación impuestos entre otros.

## 1.3 ESTRUCTURA DEL RESTO DEL DOCUMENTO

- Bloque del estado del arte: Este bloque se explicará las tecnologías que se usaron en este desarrollo, el entorno de trabajo como también las metodologías que se usaron en el día a día.
- Bloque del Desarrollo: En esta sección se explicará las decisiones tomadas para el diseño, el calendario estimado del desarrollo, como también el desarrollo de una de las tareas que se llevó acabo. Además de ello se hará un análisis de las posibles tecnologías que se usaran para la migración de la aplicación de SIGEFI.
- Bloque de Finalización: Dara a conocer la conclusión del alumno sobre toda la experiencia que ha tenido a lo largo del desarrollo. Y por último un apartado de trabajo futuro que se realizara en la aplicación.

# 2. ESTADO DEL ARTE

## 2.1 METODOLOGÍAS

### 2.1.1 AGILE Y SCRUM

Agile es un conjunto de metodologías que fueron desarrolladas para mejorar y agilizar el desarrollo de un proyecto de software. Para ello se utilizan distintas buenas prácticas que permiten un desarrollo incremental e iterativo, muchos de estas metodologías nos ayudan a minimizar errores, mejorar la comunicación de los equipos implicados, se reducen los tiempos de entrega, se mejora la calidad, y genera implicación en el proyecto de parte de cada integrante del equipo.

Algunas de las metodologías más usadas son Scrum, XP y Kanban.

- ✓ **XP**: Llamada también programación extrema, el objetivo es satisfacer al cliente. Suele ser aplicado en desarrollos de Software que suelen cambiar mucho y los requisitos que se toman al inicio del desarrollo no son precisos por lo que deben estar en todo momento obteniendo feedback entre el personal implicado y el cliente.

- ✓ **Kanban**: Es una de las metodologías que se usaron en este proyecto, su objetivo es simplificar la planificación y como estas se deben de asignar. Para ello se usa un tablero donde se plasma el flujo de trabajo (“To Do”, “In progress”, “Done”) son las columnas estándar, aunque cada equipo puede adaptar el KANBAN a su proyecto. Con esta representación del trabajo se puede controlar el avance de los proyectos.

- ✓ **Scrum**: Es una metodología enfocada en desarrollos complejos que necesitan obtener un resultado inmediato. Para ello se realizan entregas cortas (Sprints). Para poder llevar este control y entregar desarrollos de forma rápida y de calidad

  Scrum necesita las siguientes integrantes:

  - **Stakeholders**: Es el cliente quien define los requerimientos que necesita y aporta siempre feedback a nuestros desarrollos.

  - **Product Owner**: Son los encargados de estar en contacto con los Stakeholders y los encargados de que las características correctas estén en el backlog que representen a los usuarios y clientes del producto.

  - **Scrum Master**: Es el administrador del proyecto, encargado de que se sigan las reglas de la metodología y que todos los miembros del equipo cuenten

con las herramientas para desarrollar las tareas, también es el encargado de hacer las reuniones y quien realiza la planeación y liberación del producto.

- **Scrum Team**: Son los encargados de desarrollar los requisitos del backlog.

- **Reuniones Diarias**: Las reuniones diarias de los equipos agiles, tienen como misión que cada integrante explique qué es lo que está haciendo, que tareas ha completado, problemas que ha detectado, o solicitar más trabajo, todo esto en una reunión de 15 minutos, con ello se trata de aprovechar esos 15 minutos para que cada integrante informe que es lo que está haciendo. Estas reuniones ayudan al equipo ya que, si se detecta que alguien tiene dificultades en alguna tarea, el equipo intentara de ayudar a esta persona o se le puede reasignar la tarea a alguien con más experiencia dependiendo de la importancia de la tarea.

- **Reuniones retrospectivas**: El equipo se reúne al final de Spring para analizar que ha sucedido en el sprint si se ha alcanzado el tiempo estimado o no. Para ello el equipo reflexiona y analiza cómo reducir y mejorar los tiempos de desarrollo o como podría enfrentar situaciones que se han dado en el Spring finalizado de otra manera para que no afecte el desarrollo del equipo.

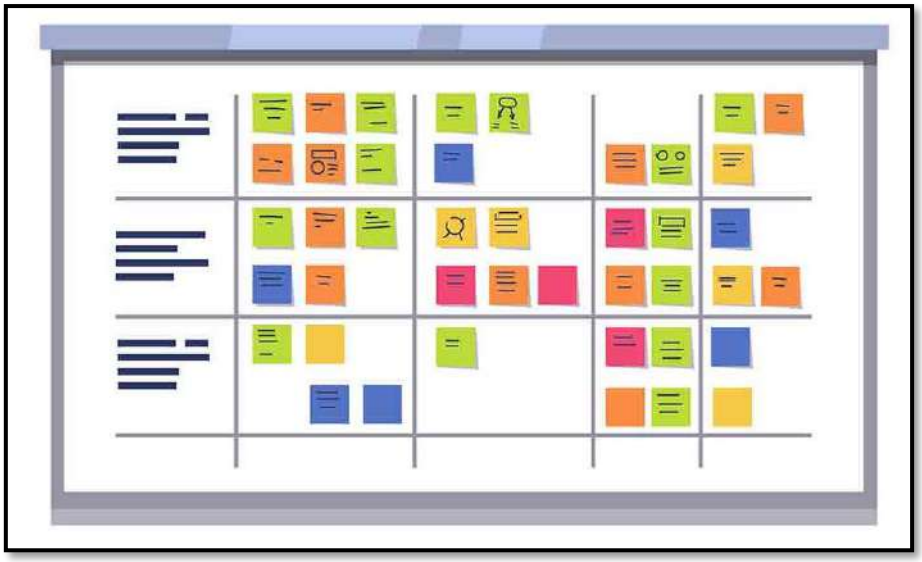


*Figura 1 Representación Scrum Board*

### 2.1.2 JIRA

Jira es también una herramienta diseñada y enfocada en el desarrollo de software y la administración de proyecto agiles, cuenta con tableros Scrum / Kanban para para el manejo de equipos agiles. Además, cuenta con tableros de incidencias por clientes (atención clientes), distintos reportes detallados de los evolutivos que se están llevando, como también estimación de tareas, feedback en tareas, estimaciones, estados de tareas, diagramas Gantt e integración con un gran número de aplicaciones en el mercado. También cuenta con distintas configuraciones de Dashboards para el seguimiento de nuevos incidencia o ver en qué estado se encuentra la incidencia y su nivel de criticidad, como también controlar el tiempo estipulado en la tarea con respecto al tiempo que le está llevando desarrollarla el programador o funcional.

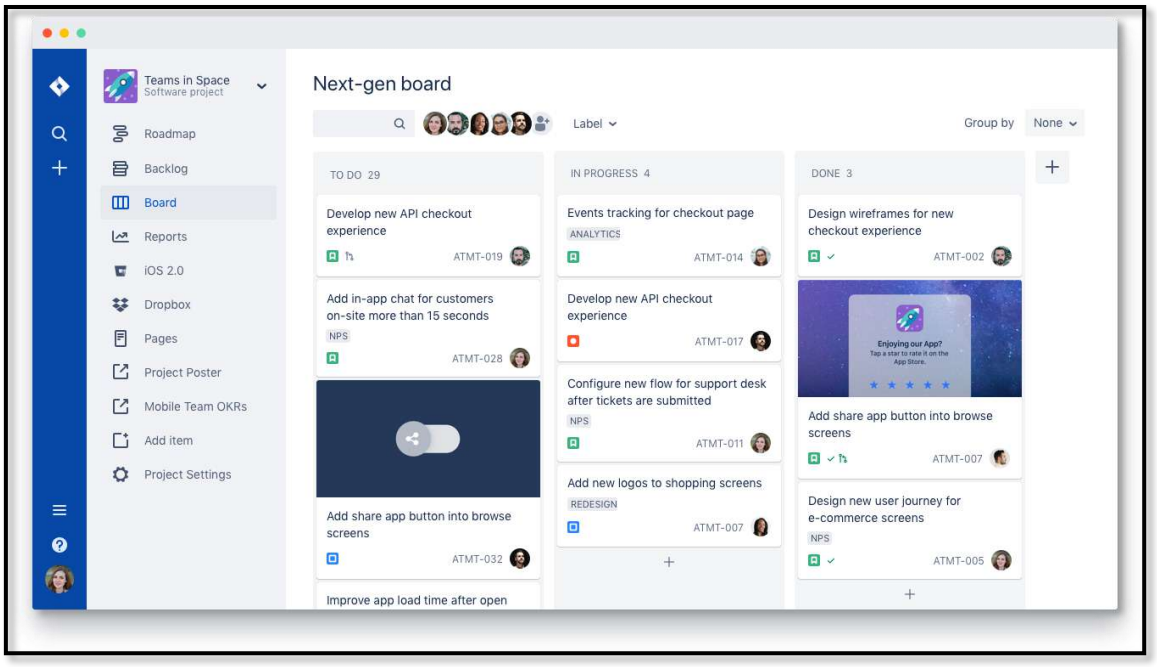



*Figura 2 Representación JIRA*

## 2.2 TECNOLOGIAS USADAS:

### 2.2.1 WEB DYNPRO

Es la tecnología estándar de SAP para el desarrollo de aplicaciones Web en el entorno SAP.

La tecnología Web Dynpro Java se compone de un entorno de tiempo de ejecución (servidor SAP J2EE de SAP) y un entorno gráfico de desarrollo con herramientas específicas de Web Dynpro (NetWeaver Developer Studio). El NetWeaver Developer Studio (NDS) provee de integración completa con el sistema de control de versiones Design Time Repository (DTR) y el sistema de transportes Java Change Management System (CMS). El servidor SAP J2EE proporciona soporte en tiempo de ejecución a las relaciones de dependencia establecidas en la NetWeaver Development Infrastructure (NWDI). La utilización del NetWeaver Developer Studio con sus herramientas declarativas y gráficas reduce significativamente el esfuerzo de implementación y mantenimiento de las aplicaciones, además de orientar el desarrollo a un proceso de diseño estructurado.

### 2.2.2 JAVA DICTIONARY

De forma análoga al diccionario de datos ABAP, el Java Dictionary sirve para declarar tipos de datos en Java asignables a objetos. Dentro del roadmap de la plataforma.

### 2.2.3 CAF

Es una tecnología propietaria de SAP que se basa en el estándar EJB para ofrecer objetos de negocio que se construyen de manera declarativa. Es una buena práctica desde SAP CE 7.11 para la construcción de la lógica en SAP NetWeaver Java.

### 2.2.4 WEB SERVICES

El servidor SAP J2EE cuenta con un servicio de ejecución y configuración de Web Services, así como con un servidor UDDI integrado. Dentro del roadmap de la plataforma.

### 2.2.5 J2EE ESTÁNDAR

Es el estándar para el desarrollo de aplicaciones empresariales sobre plataforma Java. En el lado de la vista (interfaz gráfica de usuario) J2EE utiliza JSPs (Java Server Pages) para generar las pantallas. Una JSP es una página HTML con contenido parcialmente dinámico proporcionado por la inserción en el lado del servidor de secuencias de comandos escritas en Java (scriptlets: código java incrustado en una página JSP), que permiten la ejecución de código Java y el acceso a variables de sesión como objetos bean (objetos Java reducidos a atributos de clase y métodos getter y setter para asignar y

obtener el valor de los atributos), además de proporcionar directivas JSP específicas que sirven para importar clases, incluir otras páginas dentro de la página actual, etc.

Para funciones de control en la aplicación, en J2EE se utilizan los servlets. Un servlet es un objeto Java que es capaz de recibir peticiones HTTP y generar respuestas HTTP dentro del Contenedor Web del servidor J2EE.

Para el modelo de datos, el estándar J2EE considera una buena práctica el uso de Enterprise Java Beans (EJBs). Los EJBs son objetos de negocio complejos que permiten invocación remota evitando la complejidad de uso directo de RMI o CORBA (Remote Method Invocation y Common Object Request Broker Architecture: dos tecnologías para el acceso remoto a objetos en arquitecturas distribuidas) y residen en el contenedor EJB, que gestiona su ciclo de vida. Existen tres tipos de EJBs. Los EJBs de sesión proporcionan lógica de negocio a través de sus métodos públicos, los EJBs de Entidad replican estructura y contenido de base de datos y ejercen como motor de persistencia. El tercer tipo de EJBs, los Message Driven Beans (MDBs), están orientados a la ejecución de procesos asíncronos en el servidor J2EE a través de mensajes JMS (Java Message Service).

El desarrollo con JSPs permite una máxima flexibilidad a la hora de desarrollar interfaz gráfica de usuarios, pero no ofrece más ayudas al desarrollo que librerías de etiquetas u hojas de estilo. El uso de servlets controladores permite también máxima flexibilidad a la hora de generar respuestas HTTP (control de cabeceras HTTP, generación de distintos tipos MIME : Multipurpose Internet Mail Extensions) y recibir peticiones HTTP.
Esta tecnología está integrada con el sistema de control de versiones y transportes de SAP.

Como principal inconveniente, la tecnología J2EE estándar requiere mayor esfuerzo de implementación que Web Dynpro Java, al no tratarse de programación declarativa, y requiere uniformar manualmente el diseño de la aplicación a través de hojas de estilo. Tampoco ofrece por estándar ayudas para el acceso a SAP o Web Services desde una aplicación. Por lo tanto, requiere un tiempo de construcción significativamente más alto. Además, la orientación Model View Controller (MVC: patrón de diseño base en la arquitectura de software actual) debe diseñarse específicamente.

### 2.2.6 PORTAL DEVELOPMENT KIT (PDK)

Fue la primera tecnología propia de SAP para el desarrollo de aplicaciones web Java. Basándose en un modelo propio de portlets (aplicaciones de portal), el PDK generaba aplicaciones de portal sólo ejecutables bajo el iView Runtime (entorno de ejecución de iviews, es decir, el SAP NetWeaver Portal). La vista se implementa en JSPs, como en J2EE estándar, utilizando una librería de etiquetas propia (HTMLB) que genera un diseño (Look&Feel) idéntico al estilo del portal. Como las JSPs estándar, la flexibilidad del diseño de la vista es total. El control de las aplicaciones se realiza con una implementación

propia de servlets (JSPDynPage). La orientación tecnológica del modelo de negocio es abierta: se puede implementar con cualquier API aunque, teniendo en cuenta que está orientado a aplicaciones sencillas, predomina el uso de Java Beans sencillos (clases con atributos y métodos getter y setter) que se almacenan en la sesión HTTP.

Esta tecnología está integrada con el sistema de control de versiones y transportes de SAP.

Como principal inconveniente, al igual que las aplicaciones J2EE estándar, el PDK requiere un mayor esfuerzo de desarrollo que WDJ. El desarrollo con PDK también tiene el inconveniente de que está orientada a aplicaciones pequeñas y sencillas, por su estructura de aplicación y modelo basado en un controlador único que recibe los eventos de todas las JSPs. Aplicaciones más ambiciosas complican excesivamente el control y mantenimiento del desarrollo.

### 2.2.7 MYBATIS

MyBatis es un framework de persistencia que soporta SQL, procedimientos almacenados y mapeos avanzados. MyBatis elimina casi todo el código JDBC, el establecimiento manual de los parámetros y la obtención de resultados. MyBatis puede configurarse con XML o anotaciones y permite mapear mapas y POJOs (Plain Old Java Objects) con registros de base de datos.

### 2.2.8 SPRING

Spring framework utiliza varias técnicas nuevas, como la Programación Orientada a Aspectos (AOP), el Objeto de Java Antiguo (POJO) y la inyección de dependencia (DI), para desarrollar aplicaciones empresariales, eliminando así Las complejidades involucradas en el desarrollo de aplicaciones empresariales utilizando EJB, Spring, es un marco ligero de código abierto que permite a los desarrolladores de Java EE 7 crear aplicaciones empresariales simples, confiables y escalables. Este marco se enfoca principalmente en proporcionar varias formas de ayudarlo a administrar sus objetos comerciales. Es el desarrollo de aplicaciones web mucho más fácil en comparación con los marcos de Java clásicos y las interfaces de programación de aplicaciones (API), como la conectividad de base de datos Java (JDBC), las páginas JavaServer (JSP) y Java Servlet.

## 2.3 ARQUITECTURA WEB DYNPRO JAVA

### 2.3.1 ARQUITECTURA DE UN COMPONENTE WEB DYNPRO JAVA

La unidad de desarrollo en Web Dynpro Java es el Componente. Los componentes Web Dynpro permiten estructurar aplicaciones Web complejas mediante el desarrollo de entidades reutilizables. Los componentes Web Dynpro se dividen en otras unidades relacionadas con la gestión de la interfaz de usuario, el control de navegación y el control de la aplicación y vínculo con el modelo.

Las entidades relacionadas con la interfaz de usuario son la **Ventana** (*Window*) y la **Vista** (*View*). La **Ventana** es el contenedor de vistas. Sólo en vistas embebidas en ventanas se establecen los enlaces de navegación (*navigation links*), así como los puntos de entrada (*inbound plug*) y de salida (*outbound plug*) de cada vista. La **Vista** describe el comportamiento y layout de una página visualizada en el cliente. Cada página web puede estar compuesta por una o más vistas. La vista es la entidad que contiene elementos de interfaz de usuario: campos de entrada, tablas o botones.

Los **Controladores** pueden ser de tres tipos: **controlador de componente**, **controlador de vista** y **controlador custom**. Cada uno de los tres tipos de controlador cuenta con su propio **contexto**. Los controladores de componente se encargan de acceder a los modelos e implementar el código necesario para dar funcionalidad a las vistas. Dentro de un controlador, el **Contexto** almacena los valores de las variables mediante una estructura jerárquica.

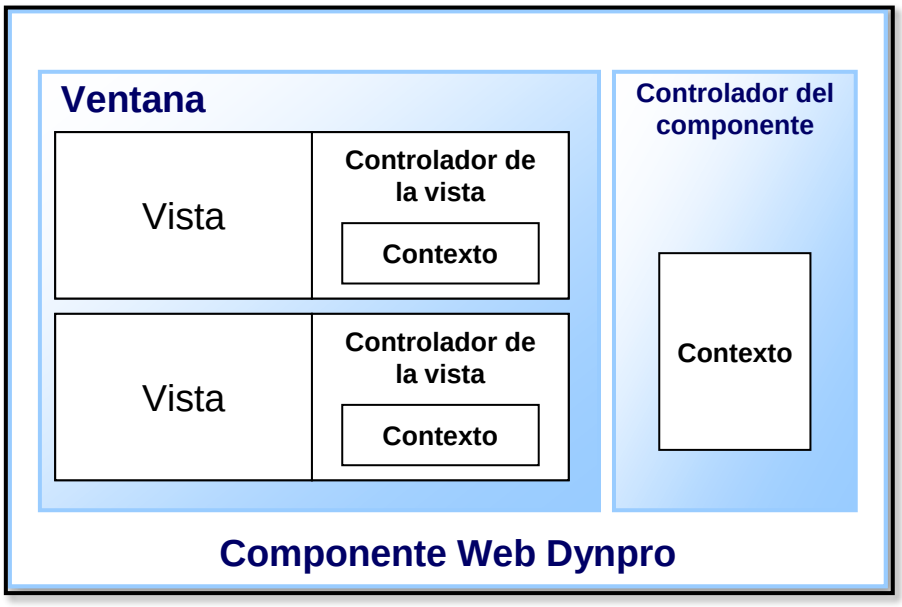



*Figura 3 Estructura de un componente Web Dynpro*

Los controladores de vista se encargan de gestionar los layouts de la aplicación, su contexto de datos y sus métodos capturadores de eventos, así como plugs de entrada y salida, funciones, internas, etc. Los datos con que los elementos de la vista son informados se almacenan y gestionan en el contexto, lo que permite representarlos o ser utilizados en la pantalla y ser accesibles desde los controladores. El controlador de la vista puede contener métodos de recuperación de datos o para el procesamiento de las entradas del usuario (chequeo de campos, por ejemplo).

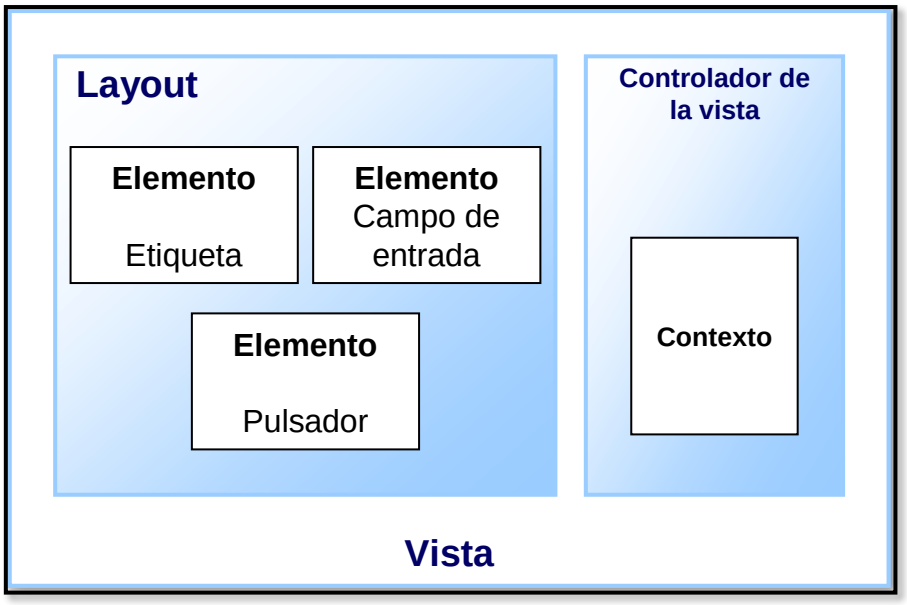



*Figura 4 Estructura de una vista*

La navegación entre las distintas vistas es posible gracias a los **Plugs**. Existen dos tipos:

- **Entrada** (*Inbound*) Definen los posibles puntos de inicio de una vista.

- **Salida**. (*Outbound*) Definen un punto de salida de la vista hacia otras vistas.

Los **Plugs** forman parte del controlador de una vista. Cada plug está asignado exactamente a una. Los Plug se asignan a vistas embebidas dentro de una ventana. Si una vista se embebe en otra ventana, se podrán definir otros plugs de entrada y salida. Para navegar de una vista a otra, cada plug de salida debe estar enlazado con uno (y sólo uno) plug de entrada. El elemento que establece esta relación es el **enlace de navegación**. Un plug de entrada puede estar relacionado con varios plugs de salida

Las variables definidas en el **Contexto** de un controlador pueden estar referenciadas a otros controladores. La referencia entre distintos contextos se denomina **mapeo del contexto**. Esto permite compartir atributos comunes entre los distintos controladores y el traspaso automático de valores del contexto de un controlador a otro. El mapeo del contexto puede existir entre controladores del mismo componente web dynpro: es el **mapeo interno**. Y también entre contextos de controladores de distintos componentes web dynpro a través de las interfaces o **mapeo externo**.

Los valores de los elementos de interfaz de usuario que permiten la entrada de datos deben estar conectados a atributos del contexto del correspondiente controlador. Es el **transporte de datos** y permite el transporte automático entre los elementos de la interfaz de usuario y los atributos del contexto.

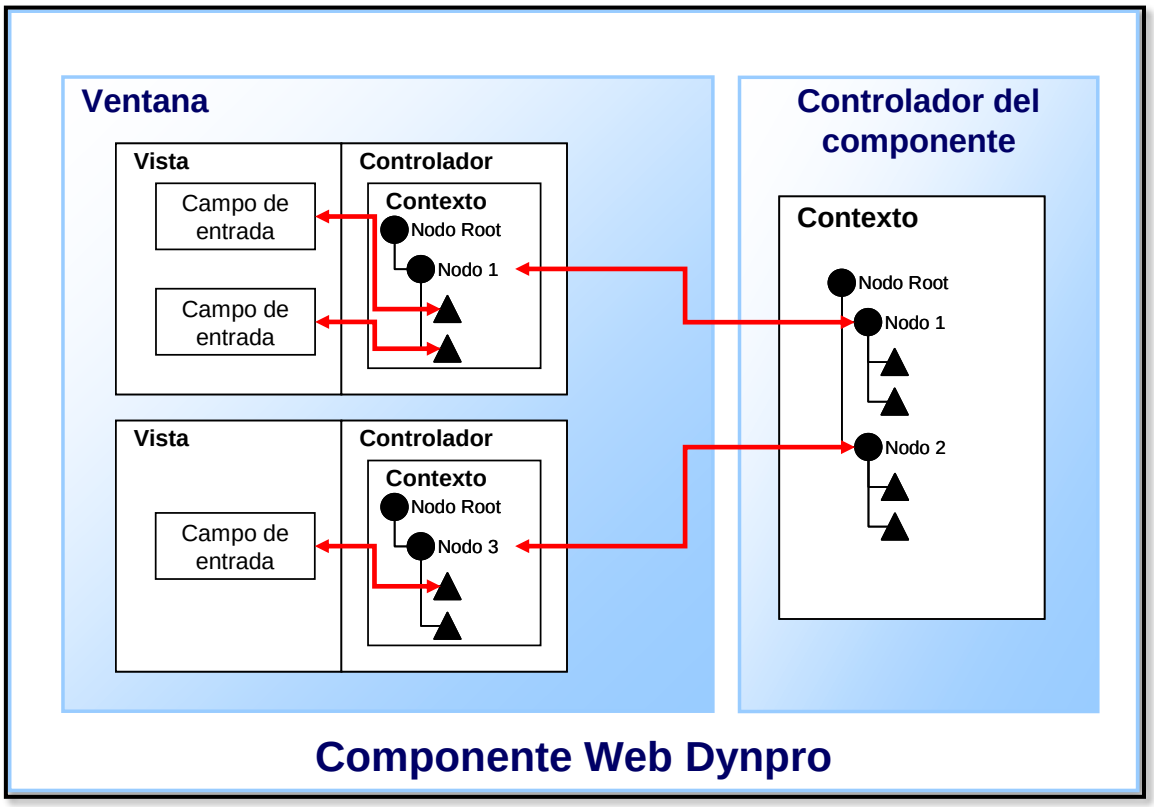



*Figura 5 Contexto y transporte de datos*

## 2.3.2 EL CONTEXTO DE UNA APLICACIÓN WDJ

Cada controlador tiene una estructura jerárquica de entidades, llamadas nodos y atributos, que utiliza para el almacenamiento temporal de datos y que es conocida como contexto. Los datos almacenados en el contexto pueden existir durante el periodo de vida del controlador (*lifespan framework_controlled*) o durante el período de vida de una vista (*lifespan when_visible*). La estructura de un contexto, normalmente, se define en tiempo de diseño, aunque también puede hacerse en tiempo de ejecución.

- **Cardinalidad de los elementos del contexto.**

  El número posible de elementos permitidos para un nodo del contexto está determinado por su cardinalidad. La incorrecta determinación de la cardinalidad de un nodo puede provocar errores en tiempo de ejecución. Los valores posibles son:

  - **1..1** Exactamente un elemento de contexto permitido. Esto significa que, en el momento de la inicialización del controlador, el contexto tiene que tener exactamente un elemento para el nodo.

  - **0..1** Uno o ningún elemento de contexto permitidos.

  - **0..n** Cualquier número de elementos de contexto permitidos.

  - **1..n** Uno o más elementos de contexto permitidos.

☑ Si a un nodo de contexto se intenta insertar un mayor número de los elementos que los permitidos, provoca un error en tiempo ejecución.

☑ Los valores 1..n y 0..n ocupan más recursos del sistema que el resto y por lo tanto sólo serán asignados cuando sean realmente necesarios.

☑ El valor 0..n tiene sentido cuando se pueden borrar elementos del nodo. Si la cardinalidad se fija en 1..n, al borrar el último elemento asignado a ese nodo se provoca un error en tiempo de ejecución.

☑ Cuando se determinan valores que comienzan en 1, el nodo contiene por defecto un primer elemento en la ejecución. Con valores que comiencen en 0, el nodo no contiene ningún en elemento por defecto en la ejecución.

- **Lead Selection**

  Una colección de elementos de un nodo puede ser accedida mediante un índice de números naturales. Se denomina Lead Selection al elemento de la colección de elementos del nodo seleccionado vía índice para su tratamiento.

- **Singleton**

El contexto es una estructura jerárquica por lo que permite la dependencia entre nodos, es decir, que los elementos contenidos en un nodo puedan estar delimitados o relacionados por un elemento de un nodo superior (**Figura 6**).

Web Dynpro sigue el principio de Lazy Data Instantiation. Esto significa que la instancia del objeto no se crea hasta que no se necesita. Según este principio, el Framework de WD no crea automáticamente todas las colecciones de los nodos dependientes para cada uno de los elementos de la colección del nodo superior. Las instancias para los nodos dependientes se crean cuando el elemento del nodo superior es el Lead Selection.

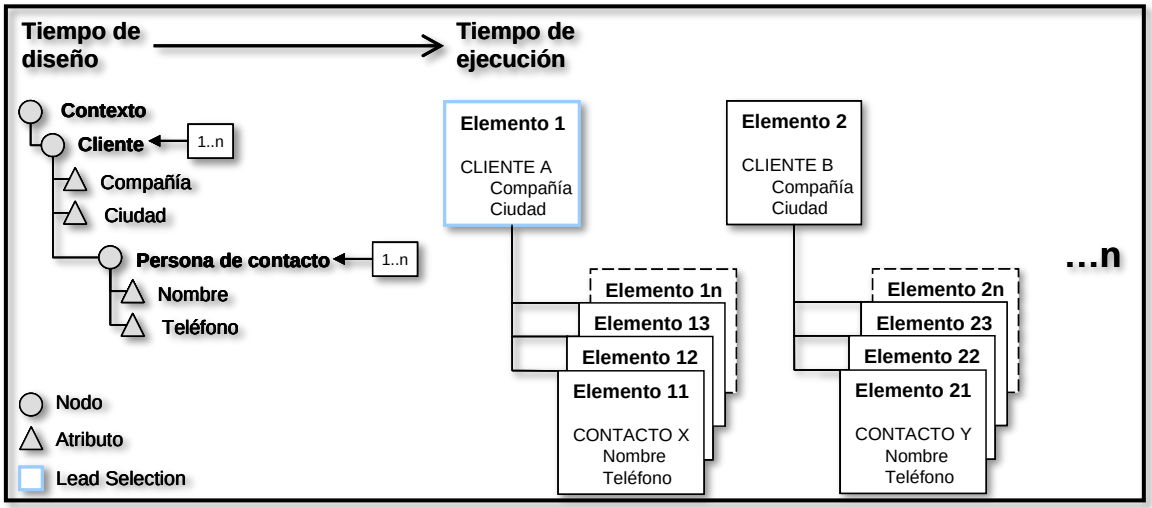



*Figura 6 Elementos de nodo dependiente de otros elementos de nodo*

## 2.3.3 DESARROLLO CON WEB DYNPRO

Web Dynpro es la propuesta de SAP para el desarrollo de aplicaciones Web desde la versión SAP WAS 6.30 hasta la actualidad.

Las aplicaciones Web Dynpro no permiten al programador definir hojas de estilo o componentes HTML a medida, sino que trabajan con una interfaz gráfica de tipo drag&drop que el propio SAP J2EE convierte en HTML. Así que los componentes gráficos que se pueden incluir están limitados a los que proporciona el propio entorno Web Dynpro, y el control sobre el aspecto de las aplicaciones (colores, diseño de componentes, tamaños y tipos de letra, etcétera) está limitado a los estilos comunes. La idea detrás de todo esto es que el programador no tenga control sobre el estilo general de las aplicaciones, sino que ese control está centralizado en unos CSS comunes a todo el entorno. La opción recomendada por SAP es que el punto de acceso para todas las

aplicaciones web sea SAP NetWeaver Portal, donde existe un editor de estilos (Theme Editor) como parte de los roles del administrador de portal.

Las aplicaciones Web Dynpro tienen una integración directa como iViews de SAP NetWeaver Portal, y pueden utilizar el EPCF (Enterprise Portal Client Framework) para comunicación entre iViews.

Las aplicaciones Web Dynpro están principalmente orientadas a generar la User Interface, gestionar el control de navegación y eventos e instanciar modelos de datos.

Pero estos modelos de negocio no se implementan directamente en Web Dynpro, sino que deben ser creados como componentes de negocio Java (Java Bean Model), ABAP (RFC Adaptive Model) o servicio web (Web Service Adaptive Model). Estos modelos se describen de forma específica en el apartado arquitectura del modelo de negocio.

### 2.3.4 MODELO DE PROGRAMACIÓN CON WEB DYNPRO

Las aplicaciones Web Dynpro están construidas utilizando técnicas de programación declarativa, programación a medida y programación dinámica basadas en el patrón de diseño Model View Controller (MVC): Modelo-Vista-Controlador.

Es decir, se puede optar por generar la IU en tiempo de diseño o generarla dinámicamente en tiempo de ejecución.

En el diseño de la aplicación se determinan los elementos de la interfaz gráfica de usuario (vista), de dónde obtienen los datos esos elementos (modelos) y cómo interactúan (navegan o, en WD, tienen establecidos navigation links) las distintas pantallas de la aplicación (controlador). Todo el código relativo a estas acciones se genera automáticamente; el único código manual necesario es el que interactúa con el backend o en la preparación de la información.

La arquitectura para desarrollo de aplicaciones Web Dynpro está basada en el patrón de diseño Modelo-Vista-Controlador que debe respetarse en la implementación de los componentes. Para ello, es importante conocer las características principales de este patrón de diseño:

- El patrón de diseño MVC trabaja en tres niveles o capas: presentación, lógica y datos.
- Las sub-tareas de una aplicación deberán estar asignadas a la capa correspondiente.
- Cada capa se comunica únicamente con la capa que es directamente su vecina.

- La comunicación entre las capas debe realizarse siempre a través de interfaces.

- Las interfaces encapsulan las tareas implementadas en cada una de las capas.

El paradigma Model View Controller (Modelo Vista Controlador o MVC) es un patrón de diseño creado originariamente para Smalltalk como sustitución del proceso Input-Processing-Output. MVC se ha extendido como patrón de diseño básico para el desarrollo de aplicaciones Web con Graphic User Interface (GUI). Inicialmente planteado para aplicaciones Cliente/Servidor, SAP para Web Dynpro han modificado en parte su funcionamiento para adecuarlo a las características del protocolo HTTP.

El objetivo de la utilización de este patrón de diseño es la creación de un sistema modular en el que los controladores actúan como conexión entre el modelo y la vista. Este tipo de arquitectura permite el desarrollo de diferentes vistas para la representación de la misma información reutilizando los componentes que procesan y suministran los datos. En un sistema desarrollado de acuerdo con el patrón de diseño MVC, los componentes individuales son independientes y pueden ser intercambiados y ampliados.

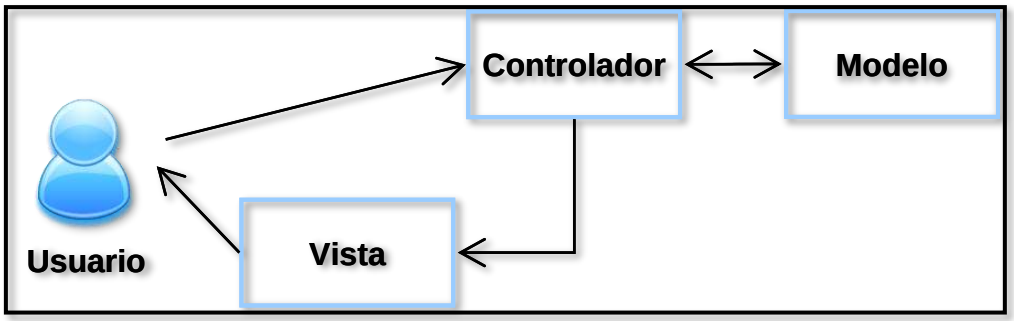



*Figura 7 Diseño Patrón MVC*

- **Modelo.** El modelo de una aplicación MVC es un componente de software integrado en la aplicación que proporciona un conjunto de datos (persistentes o generados) y los procesos de negocio (métodos) necesarios para el acceso a esos datos. El modelo en WDJ puede ser de distintos tipos: RFC, Java Bean, Web Service (de nuevo, para más información ver el apartado arquitectura del modelo de negocio). Su funcionalidad interna es una caja negra para la aplicación Web Dynpro, que los maneja de manera independiente de la tecnología; lo único importante es el manejo de su interfaz

- **Vista.** La vista es la capa que interactúa con el usuario, la apariencia gráfica de la aplicación. La vista se limita a mostrar datos al usuario y recibir eventos de éste. Para la implementación del patrón MVC, SAP ha adaptado el paradigma original (pensado para aplicaciones cliente/servidor) al protocolo en que cliente y servidor se comunican (el protocolo HTTP, es decir, un protocolo sin estado) a través de Actions (Acciones) que relacionan eventos con controladores. En Web Dynpro la vista se basa

en controles de HTMLB. HTMLB es una librería de etiquetas HTML que unifica el comportamiento y aspecto de los componentes gráficos de una aplicación web.

- **Controlador.** El controlador es un enlace entre la vista y el modelo. En otras palabras, entre el usuario y los procesos de negocio. En la implementación de Web Dynpro, no se trata de un solo controlador centralizado, sino que a la existencia de este organizador principal (Component Controller) se une un controlador propio para cada vista (View Controllers). Complementariamente, podemos crear Controladores a Medida con las mismas funcionalidades del Component Controller (los Custom Controllers), orientados a la gestión de un modelo de datos concreto.

## 2.4 EL ENTORNO DE DESARROLLO

### 2.4.1 NETWEAVER DEVELOPER STUDIO (NDS)

El SAP Netweaver Developer Studio es una plataforma de desarrollo e integración basada en Eclipse, orientada a la implementación de aplicaciones Java y ejecutadas en el servidor de aplicaciones J2EE de SAP y el SAP NetWeaver EP.

Eclipse nace en noviembre de 2001 en IBM como un Proyecto de Código Abierto con Common Public License. La idea de Eclipse es sencilla: proporcionar una plataforma básica orientada al desarrollo web con Java, sobre la cual se añaden las necesidades específicas de la tecnología utilizada. Las nuevas funcionalidades se desarrollan como plugins de la aplicación, y ocupan un lugar físico específico en su instalación.

La arquitectura de Eclipse (***ver figura 8***) consta de:

1. El Standard Widget Toolkit (SWT): es el ensamblaje predefinido de elementos gráficos para construir interfaces gráficas de usuario en Eclipse. Está desarrollada siguiendo la premisa de utilización de componentes nativos de AWT (Abstract Window Toolkit, el API básico de Java para generar interfaces gráficas de usuario), de manera que se obtenga un estilo consistente entre las distintas plataformas. Se basa en las Java Development Tools para su implementación: el API de Java más las herramientas de desarrollo.

2. El Workbench: interface de desarrollo creada en JFace. Es el entorno de programación en sí. JFace es un conjunto de herramientas con interfaz gráfica (widgets) construido con SWT basado en la reutilización de componentes. El desarrollo de plugins (Plugin Development Environment) se realiza bajo JFace.

3. El Workspace: es el directorio de proyectos, con mecanismos de histórico y reparación de errores.

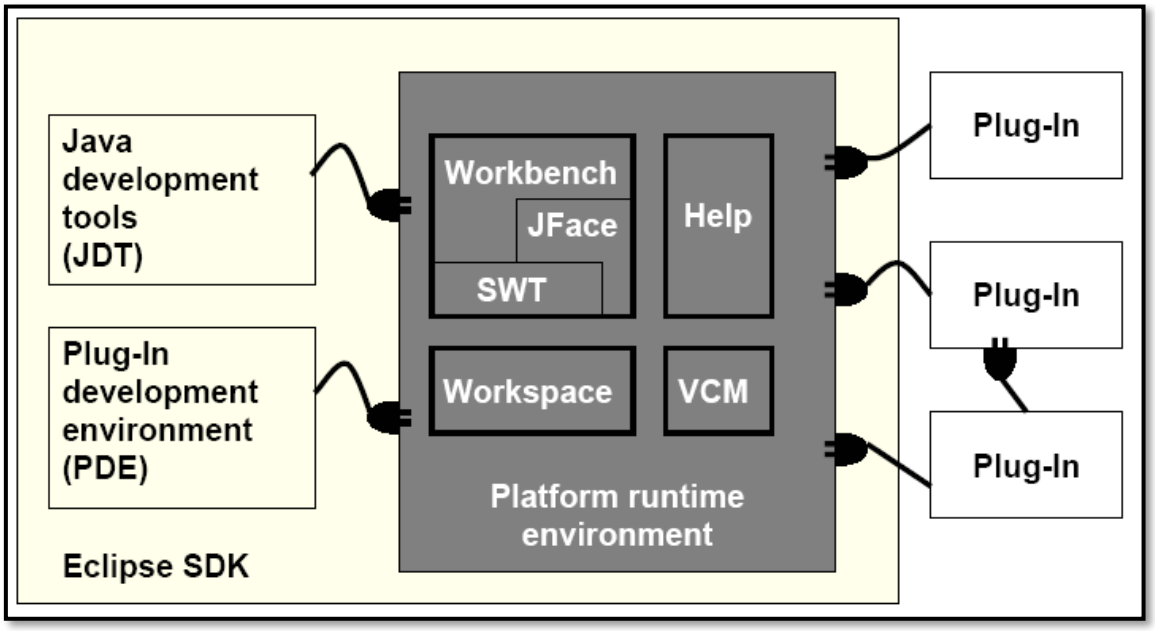



*Figura 8 Arquitectura de Eclipse*

Aparte de esta estructura, el concepto fundamental de Eclipse son las ampliaciones de la plataforma básica, es decir, los *plugins*.

El modo de trabajo en Eclipse se basa en perspectivas. Una perspectiva es un conjunto de vistas vinculadas con una tecnología concreta. Por ejemplo, la perspectiva de Java se compone de un visor de proyectos y ficheros, un editor de programación, un Outline con los atributos de una clase (variables y métodos), un visualizador de tareas para visualización de errores, warnings y tareas marcadas en el código (***ver figura 9***).

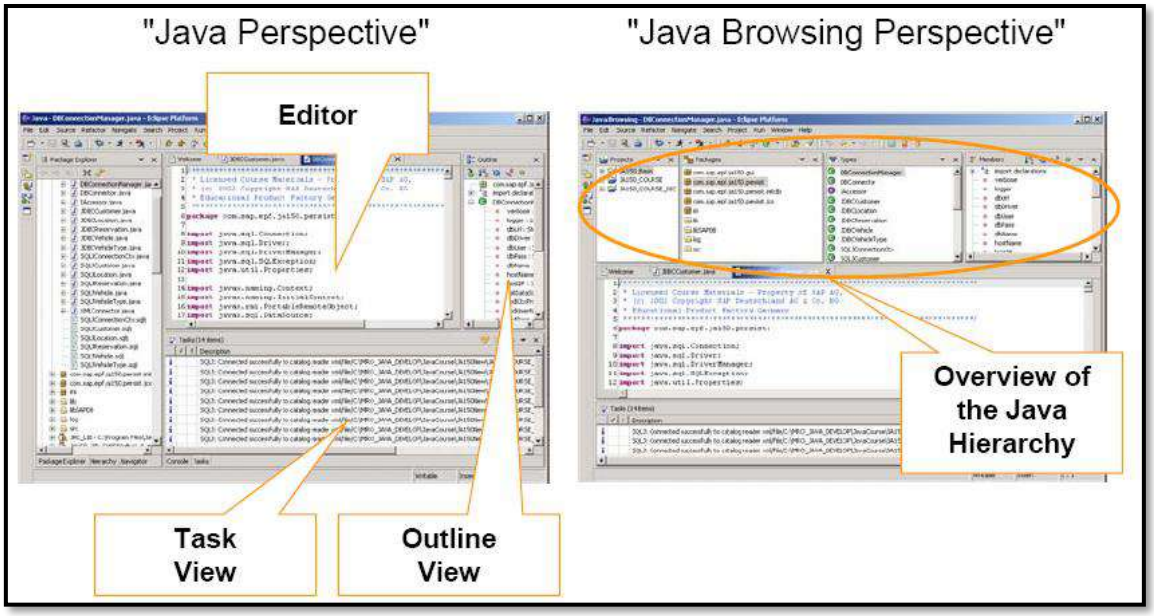



*Figura 9 Perspectivas de Eclipse*

La idea de SAP ha sido tomar como base de su entorno de desarrollo una plataforma de código abierto como Eclipse e introducir sus propios plugins que permiten el desarrollo con todas las tecnologías Java disponibles en entornos SAP (WebDynpro, SAPPortals, Mobile Engine, XI, WebServices, J2EE de SAP, Composite Environment), tal y como muestra la ***figura 10***.

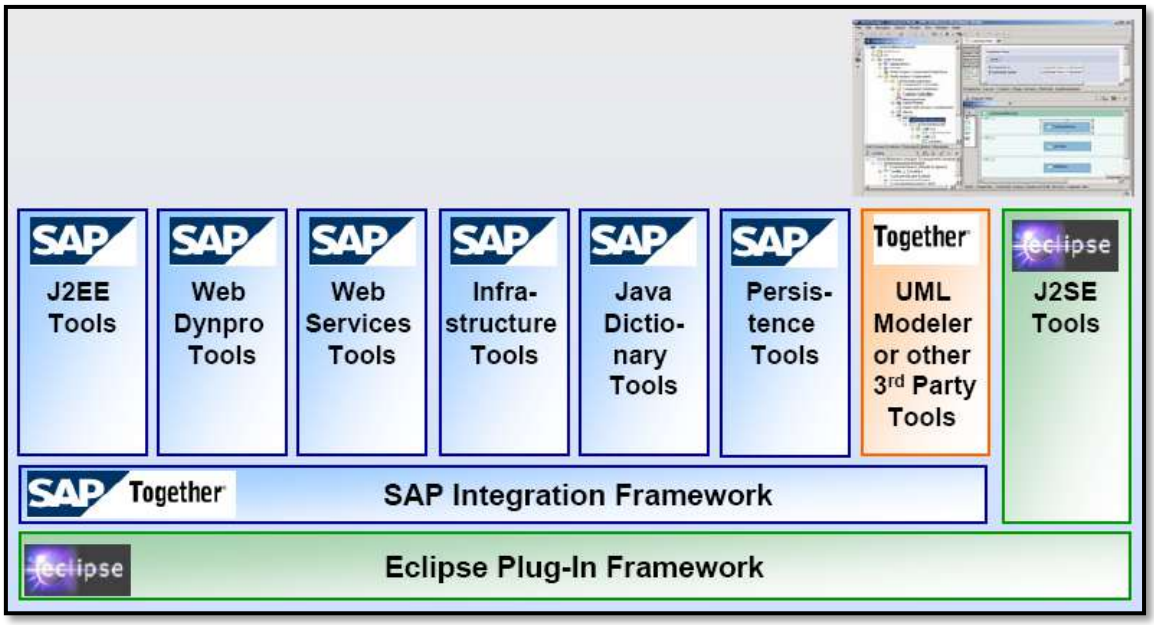



*Figura 10 Los plugins de SAP sobre Eclipse*

Además:

1. Integra la monitorización de su servidor SAP J2EE Engine desde el propio entorno de desarrollo.

2. Integra su servidor de despliegue de aplicaciones (el SDM del Web AS) para una mayor facilidad en el trabajo del desarrollador.

3. Facilita los procesos de construcción de todos sus componentes. En programación J2EE estándar se suelen utilizar scripts de Ant. Ant es una herramienta que sirve para reconfigurar la compilación y construcción de un componente. Para ejecutarlo, es necesario definir un XML con las dependencias, directorios de trabajo, etc. En NDS, las tareas de compilación y construcción son automáticas y transparentes para el programador.

4. Facilita el proceso de despliegue a través del Software Deployment Manager (SDM). El SDM es el proceso encargado de desplegar componentes en el servidor SAP J2EE.

5. Integra el entorno con su infraestructura de desarrollo (NWDI), facilitando el trabajo con:

    a. Design Time Repository (DTR): sistema de control de versiones.
    b. Component Build Service (CBS): servicio de construcción y activación de componentes en el servidor.
    c. Change Management Service: sistema de transporte

- **NWDI:** NWDI proporciona una infraestructura de desarrollo consistente para manejar el ciclo completo de las aplicaciones basadas en Java.

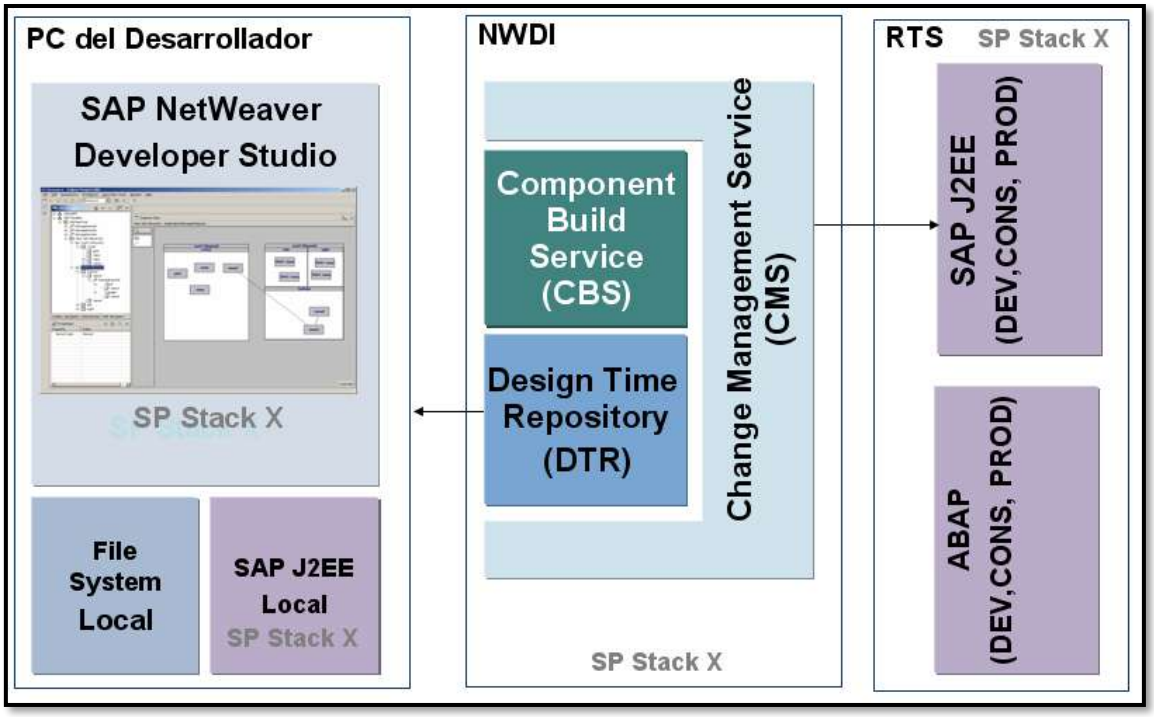



*Figura 11 Elementos de Infrestructura NWDI*

- Sirve para gestionar:
  - El proceso de desarrollo y sus entornos
    - Define desarrollo, despliegue y ejecución en distintos sistemas.
    - Objetivo: Mantenimiento de los desarrollos.
  - La estructuración de las aplicaciones
    - Una base técnica para definir claramente la estructura de una aplicación.
    - Objetivo: Reutilización de componentes.
  - Modificaciones de cliente al estándar Java
    - Proporciona las fuentes Java de las aplicaciones estándar.
    - Objetivo: Soporte de ajustes de modificación tras upgrades.

Tal y como se muestra en **figura 11**, la NetWeaver Development Infrastructure (NWDI) consta de tres componentes:

- DTR : Design Time Repository.
- CBS : Component Build Service
- CMS : Change Management System

Cada uno de los tres componentes se describe a continuación en un punto independiente.

Desde el punto de vista del sistema, los componentes fundamentales del NWDI son DTR, CBS y CMS, pero desde el punto de vista de la arquitectura de desarrollo los componentes básicos son Development Components, Software Components y Tracks.

- Development Component (DC):

Un Development es la unidad mínima de desarrollo de componentes en NWDI, la mínima unidad sobre la que se puede realizar un seguimiento de estado. Se corresponde con un proyecto en NetWeaver Developer Studio y con un componente de CBS.

- Software Component (SC):

Es la unidad de distribución dentro de NWDI. Cada aplicación se compone al menos de un SC. Un SC contiene los Development Components (componentes de desarrollo) que contienen la implementación.

Si una aplicación es compleja, es habitual que se divida en distintos SCs, ya sea por división funcional (cada SC puede ser una entrada de primer nivel de Menú que contiene *n* entradas de segundo nivel de menú, por ejemplo), tecnológico (un SC para librerías, otro para la capa de presentación, otro para la lógica de negocio) o por ciclo de vida del software (un SC para cada puesta en producción parcial).

Para determinar la ruta de transportes de cada desarrollo, los Software Components se asocian a **Tracks**.

- Track:

Un **Track** es una configuración del CMS que determina qué sistema están asociado a cada entorno de ejecución (Desarrollo, Consolidación, Test, Producción) y sobre qué Software Components se puede desarrollar en ese entorno.

### 2.4.2 DESIGN TIME REPOSITORY (DTR)

El DTR es el sistema de control de versiones del NWDI.

Se basa en un desarrollo distribuido y concurrente.

- Incluye aviso de conflictos entre versiones o usuarios
- Incluye distintas opciones para la resolución de conflictos, incluyendo el *merging* (fusión) de versiones.

Está orientado a ficheros y carpetas

- El DTR sólo maneja ficheros y carpetas
- No maneja objetos de programación

Su proceso de trabajo se basa en operaciones de Check in/check out:

- Los ficheros se editan offline mediante un *check out* del DTR. Un checkout es la descarga de un fichero del servidor de versiones (DTR) para su edición, incluyéndolo en una actividad (registro de ficheros editados actualmente por un usuario) para su transporte.
- Los ficheros se actualizan en el servidor mediante un *check in.* Un checkin supone el cierre de una actividad, finalización de edición de un fichero y subida de ese fichero al servidor de versiones (DTR).

La estructura de del DTR y un ejemplo se muestra en la **figura 12.**

En este ejemplo el desarrollador accede desde su entorno de desarrollo local en su PC a través de Eclipse a un fichero del DTR mediante los protocolos WebDAV (protocolo de acceso web a ficheros utilizado por el DTR) y DeltaV (protocolo de gestión de versiones utilizado por el DTR). Este fichero (el VR1 en el gráfico, por ejemplo) se encuentra en el workspace inactivo de desarrollo. El workspace inactivo de desarrollo es el conjunto de DCs que están en proceso de edición sin activar, es decir, la versión de las fuentes que aún no se ha compilado y consolidado en el servidor. Es el área de trabajo del desarrollador. Cuando el programador edita el fichero lo añade a una actividad (representada por un triángulo amarillo en el dibujo), la actividad *act a*

Cuando considera cerrados los cambios los activa desde Eclipse y se produce un intento de compilación de este fichero junto con el resto de las versiones activas por parte del CBS. Si la compilación es correcta, el fichero pasará al workspace activo, el que contiene las fuentes consolidados.

A partir de ese momento, si el programador realiza una nueva modificación del fichero, se generará una nueva versión del mismo en el workspace inactivo (versión 2 de VR1 en el ejemplo), quedando la versión del workspace activo intacta.

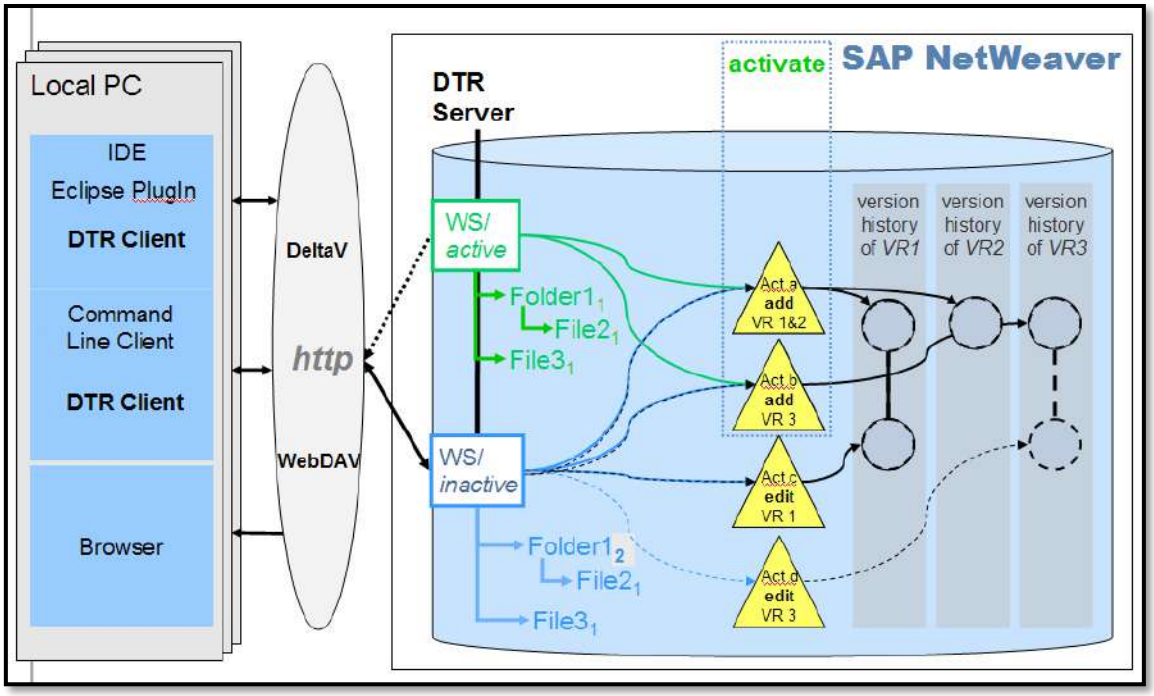



*Figura 12 Ejemplo de uso de DTR*

### 2.4.3 COMPONENT BUILD SERVICE (CBS)

El CBS es un servicio del servidor SAP J2EE para la construcción centralizada y gestión de buildspaces (espacios de construcción).

Características principales del CBS:

- Gestiona los archivos que representan cada estado de un software component.
- Proporciona el resultado de la última construcción a los desarrolladores
- Se administra vía Web UI.
- El proceso de construcción del CBS se basa en componentes, no en clases.

Funciones principales del CBS:

- La construcción de componentes a petición de desarrolladores en el momento de la activación.
- La reconstrucción automática los DCs dependientes.
- La encolación de las peticiones de construcción para ejecutarlas de forma asíncrona y ejecuta en paralelo dichas peticiones.

### 2.4.4 CHANGE MANAGEMENT SERVICE (CMS)

El Change Management Service (CMS) es el sistema integrado de transportes de NWDI para componentes Java de SAP. Representa el control del ciclo de vida del software dentro de la infraestructura de desarrollo.

Sus principales tareas son:

- Configurar el entorno (sistemas de desarrollo, consolidación, test y producción; traces; componentes involucrados en cada track).
- Controlar el transporte y despliegue de objetos de objetos de desarrollo.

De acuerdo con las dos tareas anteriores, el CMS consta de:

- Landscape Configurator
- Transport Studio

En el CMS se definen los tracks, que indicarán qué Software Components se despliegan en qué entornos de ejecución.

Cada Track (*ruta de transporte*) puede constar de cuatro entornos: desarrollo, consolidación, test y producción (**Figura 13**)

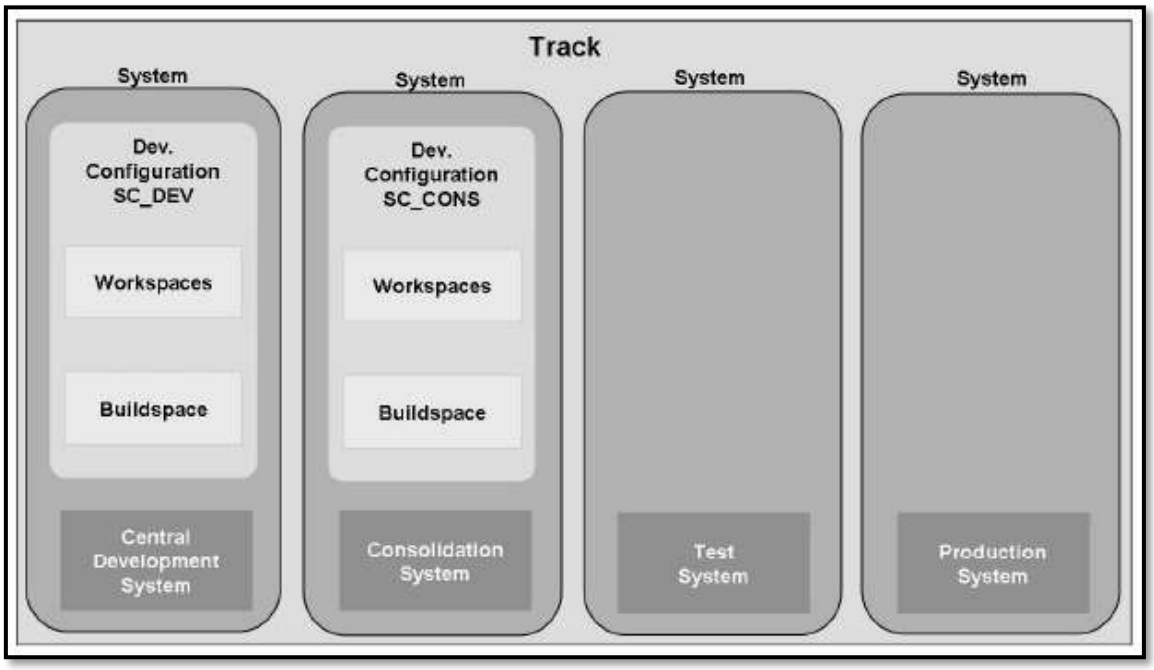



*Figura 13 Entornos de Ejecución en el CMS*

### 2.4.5 DEPENDENCIAS ENTRE DCS

Para NetWeaver Developer Studio 7.0 las dependencias entre componentes de NWDI se definen desde el nodo *Used Components* en la perspectiva Development Configurations (ver figura 5.2.1).

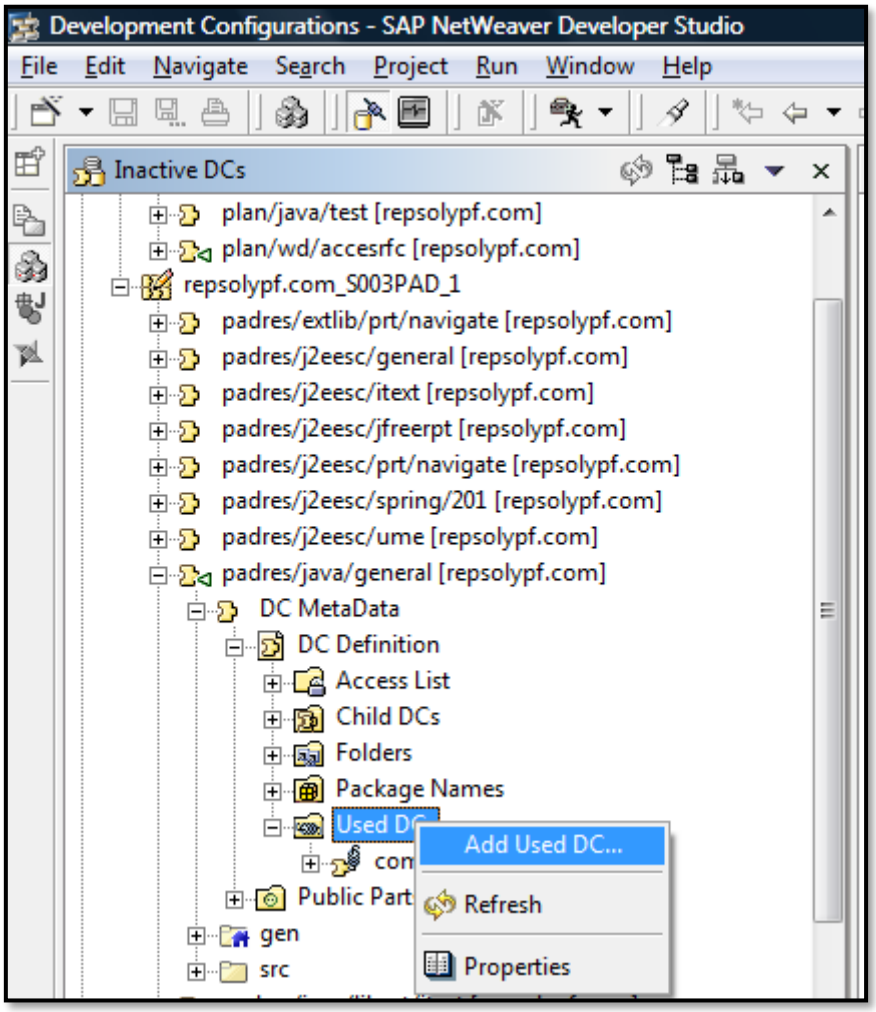



*Figura 14 Dependencias en NW*

Para desarrollo dependencias se establecen desde la perspectiva Development Infrastructure, seleccionando el componente y, en la vista Component Properties, con los botones *"Add"* y *"Remove"* .

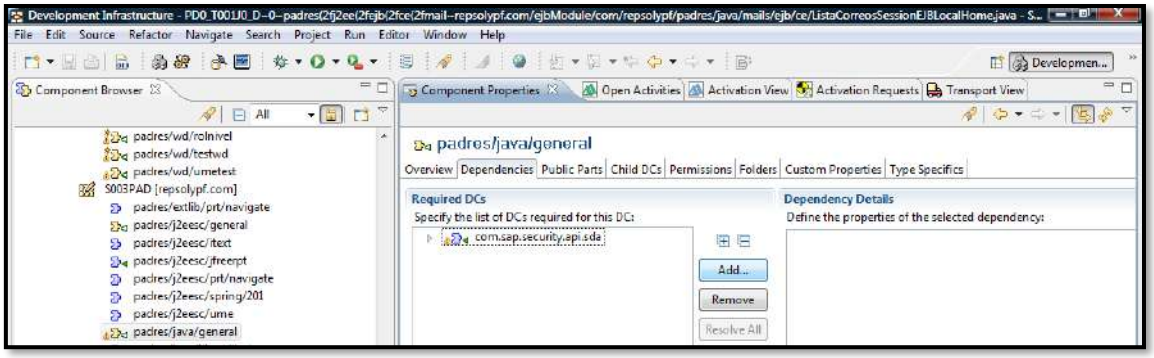



*Figura 15 Agregar y Remover dependencias*

Los tipos de dependencias entre componentes que existen son:

- *Design Time (needed for special editors only)*

  Tipo de dependencia utilizado por herramientas de modelado. Actualmente no se utiliza.

  *Necesario sólo para editores de modelado especiales.*

- *Build Time*

  Esta dependencia especifica que un componente es utilizado durante la compilación o como envolvente (wrapping) en otro componente.

  *Necesaria para compilación.*

- *Deploy Time*

  Esta dependencia especifica que el componente utilizado debe existir en el sistema de ejecución para permitir desplegarse al componente que lo usa.

  Sirve para asegurar el correcto orden de despliegue de componentes.

  *Evita el despliegue si las dependencias no se encuentran.*

- *Runtime*

  Esta dependencia especifica que un componente requiere otro en tiempo de ejecución.

Indica si un componente desplegable utiliza otro. Esta dependencia siempre se establece entre tipos de componentes ejecutables (Web Dynpro, Enterprise Application) En los proyectos de tipo Web Dynpro se debe complementar con la referencia a las librerías (**figura 5.2.1:** Library references) y aplicaciones (**figura 5.2.2:** Sharing references**)**. Ambas referencias (Library references y Sharing references) se establecen en el menú contextual *Properties* del proyecto Web Dynpro, dentro de la entrada *Web Dynpro References.*

En las aplicaciones J2EE las referencias se incluyen en el fichero application-j2ee-engine.xml. Dentro de la pestaña *General* se añaden en el cuadro References (ver **figura 5.2.3**), indicando si se

# 3 BLOQUE DE DESARROLLO

## 3.1 DESARROLLO Y ENTORNOS

Gráficamente, y de manera análoga a los procedimientos SAP, los procedimientos SAP NetWeaver seguirán el flujo siguiente (ver **figura 16**):

Desarrollo **> (Paso 1) >** Preproducción **> (Paso 2) >** Producción.

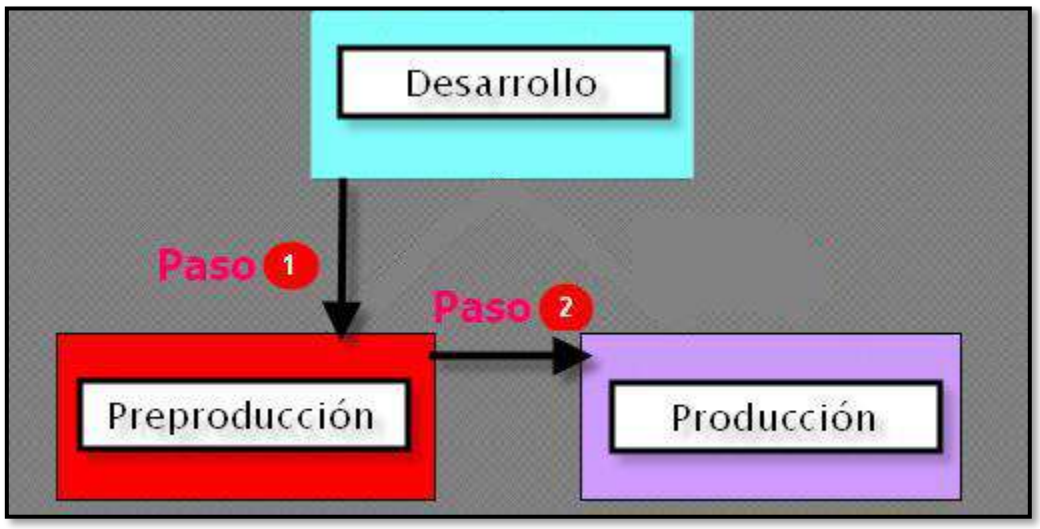



*Figura 16 Transporte entre entornos*

- El paso 1 se realizará únicamente cuando se haya aprobado la documentación de análisis, diseño técnico y pruebas y las pruebas unitarias hayan resultado satisfactorias.

- El paso 2 se realizará cuando las pruebas de Preproducción estén aprobadas en su totalidad.

Salvo autorización expresa, no está permitido crear componentes locales en los entornos de desarrollo por desvinculación del DTR. Todos los componentes deben estar asociados a un Track transportable.

No se permitirá activar actividades que no se pasen nunca a producción.

Al terminar un proyecto, igualmente, se requerirá al responsable, ya sea interno o un contratista, para que cumpla el ciclo de todas las actividades en el entorno, ya sea para transportarlas o eliminarlas.

Para mantenimientos que no sean en sí proyectos, no se permitirá tener actividades abiertas con una antigüedad superior a un mes.

## 3.2 GESTION DE ACTIVIDADES

### 3.2.1 DESARROLLADOR: ASIGNACIÓN DE MODIFICACIONES A UNA ACTIVIDAD

Cuando modifiquemos algún fichero del componente, nos pedirá autorización para realizar un Check Out de los ficheros del servidor. También se pueden editar directamente desde la perspectiva del DTR, indicando en este caso si la edición es exclusiva (se bloquea la edición de ese objeto para el usuario con el que estamos logados) o no.

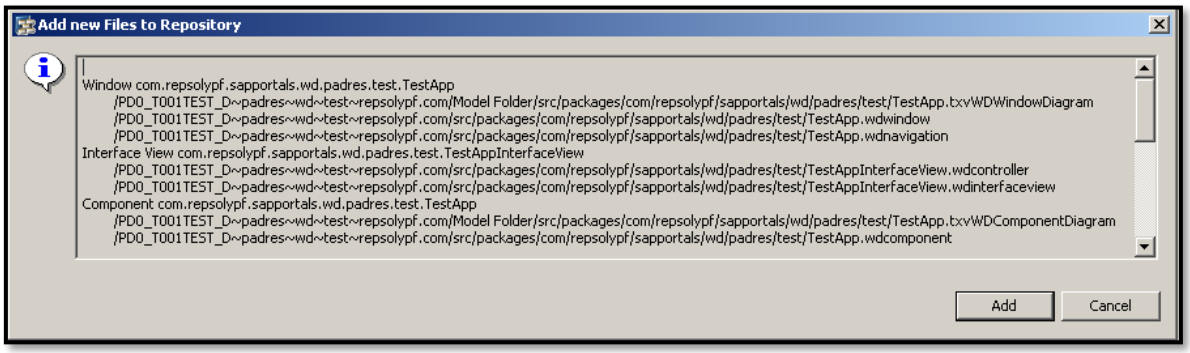



*Figura 17 Detección de modificación de un componente*

Cuando aprobamos el Check Out, nos pedirá una actividad (equivalente a una orden de transporte ABAP en SAP ECC) en la que incluir los cambios para un posterior transporte

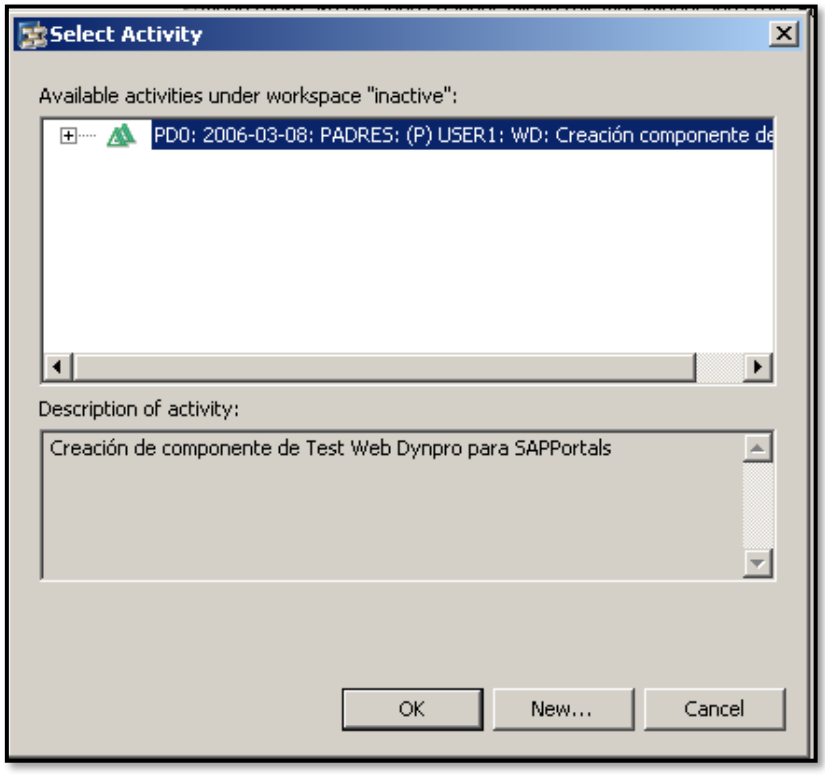



*Figura 18 Asignación de una modificación a una actividad existente*

Si no existe ninguna actividad abierta, crearemos pulsando el botón “New…”:

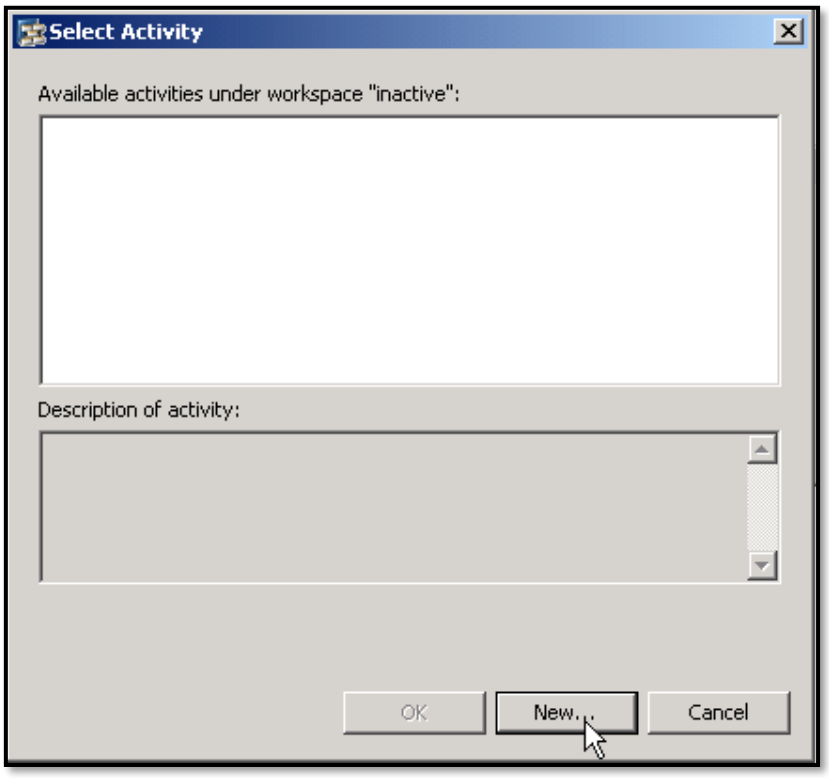



*Figura 19 Creación de una nueva actividad*

En el caso de que ya exista una o varias Actividades abiertas, seleccionamos la adecuada en la ventana anterior (**Figura 18**), si no existe una definida procedemos a crear una nueva actividad definiéndole un nombre y descripción.

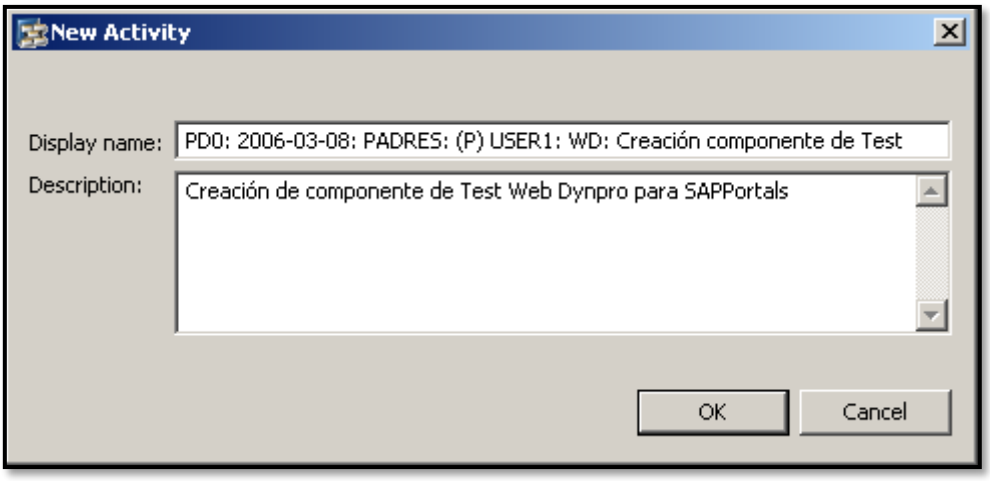



*Figura 20 Descripción de actividad*

Una vez que se han realizado los cambios, desde la perspectiva de Development Configurations podemos acceder a las Actividades abiertas (pendientes de hacer *checkin* al servidor) en la vista de "Open Activities".

*Figura 21 tabla de perspectivas NW*

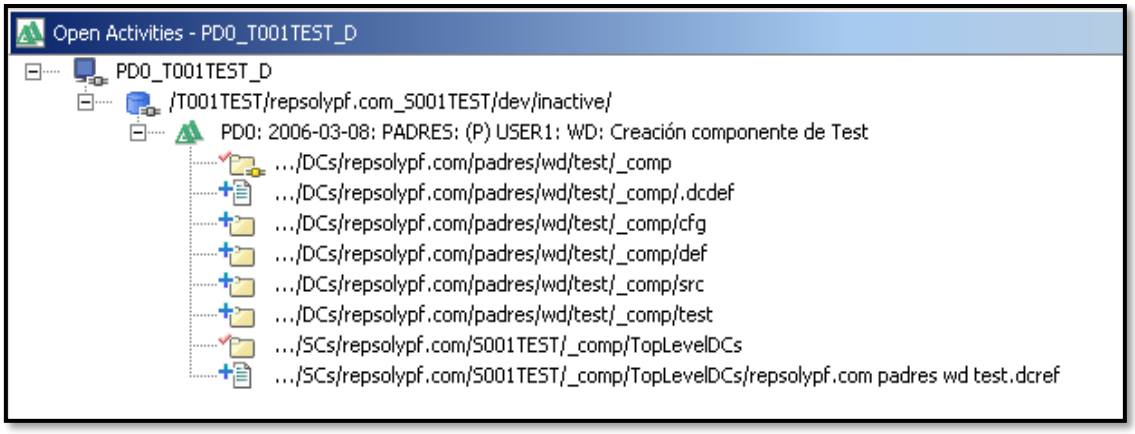



*Figura 22 Contenido de la actividad*

### 3.2.2 DESARROLLADOR: CERRAR LA ACTIVIDAD A TRANSPORTAR

Para cerrar la actividad –imprescindible antes de poder transportar-, desde la vista de "Open Activities", con el botón derecho sobre la actividad abierta seleccionamos "Checkin". En la parte inferior, podemos observar los ficheros implicados en el transporte.

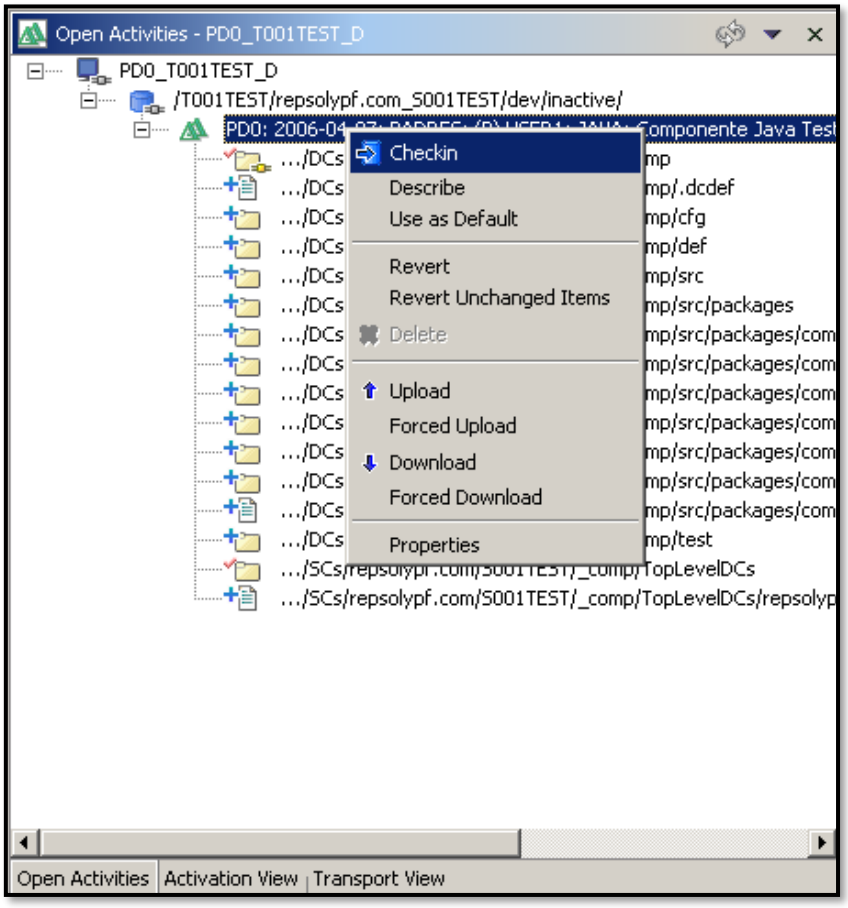



*Figura 23 Check-IN de la actividad*

El NDS solicitará un nombre, donde dejaremos establecido el nombre según los criterios descritos anteriormente. Este será el nombre que veremos desde la consola de Change Management Service.

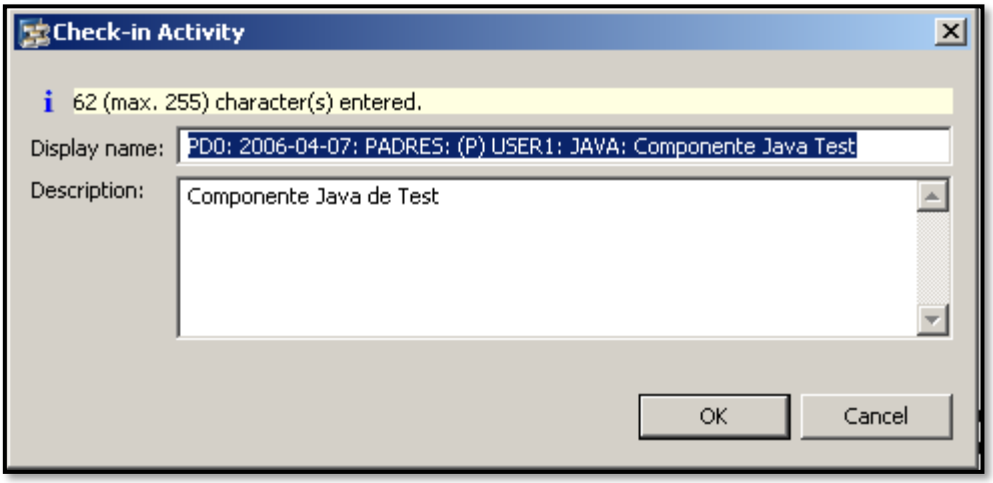



*Figura 24 Comprobación del check-in de la actividad*

### 3.2.3 DESARROLLADOR: LIBERACIÓN DE LA ACTIVIDAD

Desde la vista de Transporte del NDS (Transport View), en el nodo Waiting podemos encontrar la orden activada. Para liberarla, seleccionando con botón derecho hacemos "Release" de la actividad.

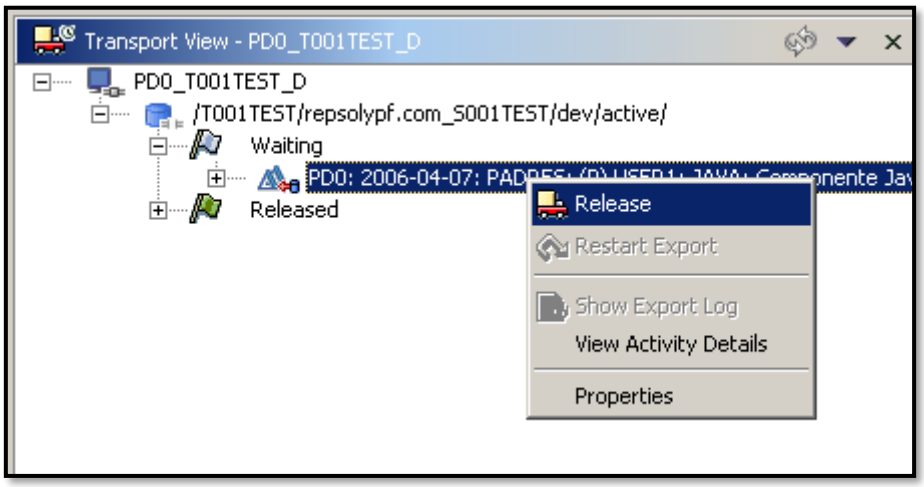
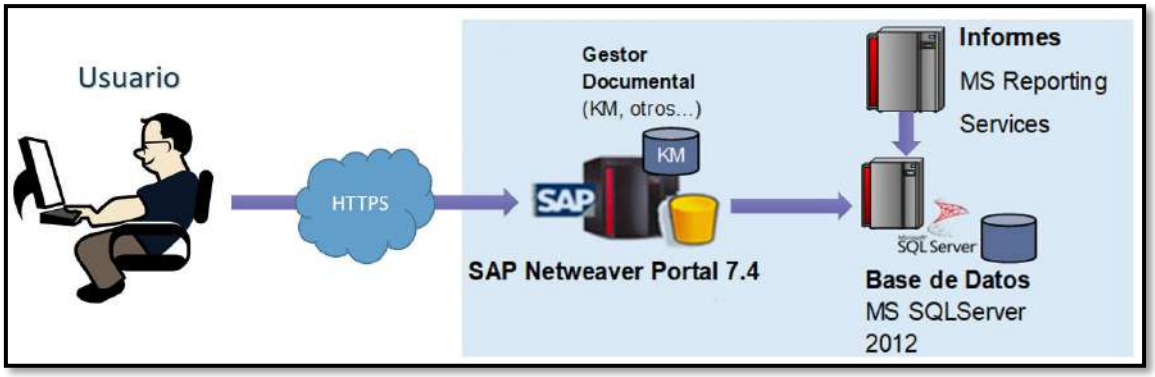



*Figura 25 Liberación del desarrollo*

## 3.3 APLICACIÓN SIGEFI

SIGEFI gestiona los modelos impositivos de ámbito nacional (estatal, autonómico y local) e internacional, siendo flexible a la hora de incorporar nuevos tributos y todas las adaptaciones Legislativas. Cubre completamente la gestión de los impuestos. La solución se basa en la plataforma SAP Netweaver, siendo compatible con cualquier solución de integración del mercado para la carga de datos desde cualquier sistema de negocio, sea del fabricante que sea. La solución completa está integrada en un Portal Web de SAP, por lo que es accesible a todos los usuarios de la red de cualquier país sin previa instalación, a través del navegador.

El sistema está implementado en una infraestructura SaaS (Software como servicio), los clientes adquieren un acceso de a la aplicación donde pueden administrar toda la aplicación, esto les brinda a los clientes un ahorro de personal técnico para mantener la aplicación, como el ahorro de mantener una infraestructura tecnología.




## 3.4 IMPLEMENTACION DEL MODULO DEL IMPUESTO SOBRE EL BENEFICIO

Este módulo permite introducir y gestionar toda la información relativa a valores y magnitudes relativos al Impuesto sobre el Beneficio Anual (en adelante, IBA) de las entidades individualmente, así como consultar la citada información y generar informes tanto de forma individual como consolidada en función de diversos parámetros.

La información relativa a los valores y magnitudes del IBA que pueden introducirse, gestionarse o consultarse pueden referirse a dos períodos siempre dentro de un mismo ejercicio fiscal o período impositivo:

- Cierre: Es la provisión del IBA que se calcula y contabiliza al cierre del ejercicio al que se refiere.
- Liquidación: Referida a la información y valores que se consignan en la declaración/liquidación que se presenta en Hacienda.

Entre otras cuestiones, se deberá informar obligatoriamente sobre los siguientes valores o magnitudes:

- El cálculo del Gasto por IS del ejercicio y movimientos en balance de Impuestos diferidos que se hayan contabilizado al cierre del ejercicio.
- El cálculo de IS corriente a pagar o devolver.
- Detalle de Deducciones o créditos fiscales generados, aplicadas y pendientes
- Detalle de BIN generadas, aplicadas y pendientes

➢ Inicio de desarrollo:

El equipo funcional situado en Madrid que tiene contacto directo con el cliente fue el encargado de recolectar todos los requisitos de los clientes para hacer frente a sus necesidades.

Una vez redactado el documento de requisitos del cliente este llega al equipo técnico que estima la cantidad de horas en total que se requiere para el desarrollo.

Se inicia el documento técnico donde se analiza las tablas que se creerán, cuales se reutilizaran, componentes nuevos que se deben de crear etc. Como también el comportamiento de la aplicación y como se comunicarán entre componentes y las dependencias que existirán.

Una vez hecho el documento técnico se realiza una estimación aproximada que tomara cada tarea y el resultado final es el siguiente la estimación inicial del proyecto.

El calendario de esfuerzo es el siguiente:

| | DIAS TOTALES DE DURACION | 77,82 |
|---|---|---|
| **Ajuste BI** | **Total en dias** | **13,97** |
| | Diseño Pantalla | 6,05 |
| | BBDD Pantalla | 2,08 |
| | Tokens(Dinamicos) | 0,83 |
| | Volcado al Historico AID | 1,67 |
| | Volcado Datos Previos | 1,67 |
| | Informe Pestaña | 1,67 |
| **BINS** | **Total en dias** | **16,35** |
| | Diseño Pantalla | 8,75 |
| | BBDD Pantalla | 1,00 |
| | Tokens | 0,83 |
| | Volcado Historico AID | 2,00 |
| | Volcado Historico AIDPID noActivos / no registrados | 0,83 |
| | Modificar Volcar Datos Previos | 1,94 |
| | Informe Pestañas | 1,00 |
| **Deducciones** | **Total en dias** | **9,19** |
| | Diseño Pantalla | 1,67 |
| | BBDD Pantalla | 1,25 |
| | Tokens | 0,83 |
| | Volcado al historico AIDPID | 1,67 |
| | Volcado Historico AIDPID noActivos / no registrados | 0,83 |
| | Modificar Volcar Datos Previos | 1,94 |
| | Informes Pestañas | 1,00 |
| **Historico AIDPID** | **Total en dias** | **16,04** |
| | Diseño Pantalla | 8,04 |
| | BBDD pantalla | 2,00 |
| | Tokens(Dinamicos) | 0,83 |
| | Volcado a AIDPID | 1,67 |
| | Volcado a inventario AIDPID | 1,67 |
| | Informes Pestañas | 0,83 |
| | Diseño Interfaz | 1,00 |
| **AIDPID** | **Total en dias** | **2,78** |
| | Modificacion Pantalla | 1,67 |
| | BBDD pantalla | 0,83 |
| | adapatacion Repsol | 0,28 |
| **inventario AIDPID** | **Total en dias** | **19,49** |
| | Mantenimiento Sociedades | 1,00 |
| | Mantenimiento de Grupos Fiscales | 1,00 |
| | Cierre Ejercicio Individual | 0,83 |
| | Cierre Ejercicio Consolidado | 2,78 |
| | Interfaces Diseño | 5,55 |
| | importacion / exportacion datos | 8,33 |
| | Pruebas Funcionales | 10,00 |
| | Pruebas Aceptacion | 10,00 |

*Figura 26 Calendario del modulo de Impuesto sobre Beneficio*

### 3.4.1 DISEÑO DEL PROYECTO

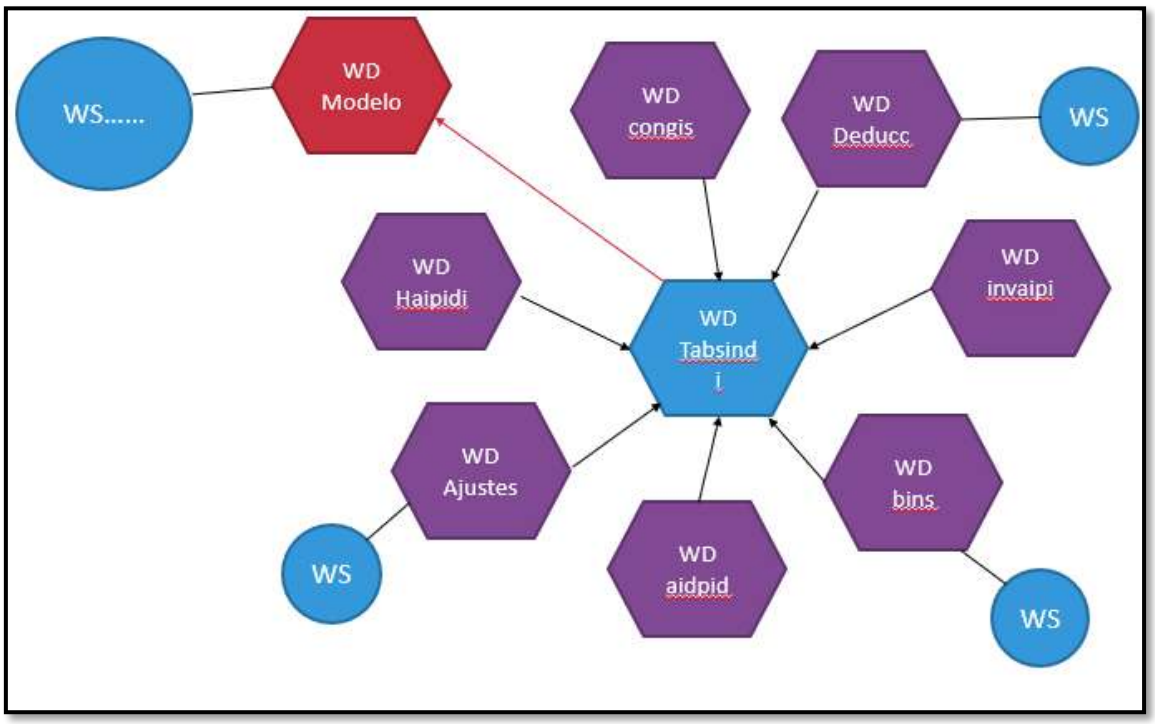



*Figura 27 Diseño del modulo Front-End*

El componente principal es el tabsindi que es el componente que muestra por pantalla la carga de una sociedad individual. Al ser el componente padre este contiene las pestañas que son subcomponentes (Figura tal). Cuando una pestaña es modificada esta puede en el mayor de los casos afectar las demás pestañas por lo que debe de notificar al componente padre que es el que notificara al componente modelo para realizar el mantenimiento.

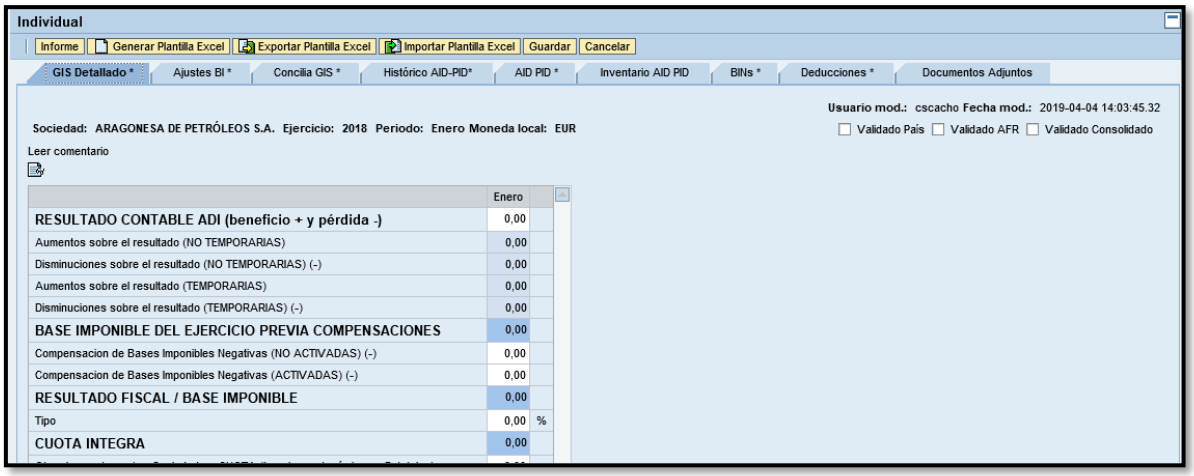



*Figura 28 Interfaz modulo de Impuesto sobre Beneficio*

El componente modelo, es el componente encargado de todo el manejo de los datos, operaciones como actualizar, obtener, eliminar, recalcular.

A continuación, se explicará el desarrollo de una de las tareas:

### 3.4.2 DESARROLLO DE LA PESTAÑA DE AJUSTESBI

Se inicia el desarrollo por la parte de la lógica de negocio, el cual el patrón DAO para acceder a la capa de persistencia:

**DAO**: Es el patrón que ofrece un componente software que actúa como un puente de implementación entre los datos que se van a almacenar en la capa de persistencia y la capa de lógica de nuestro sistema.

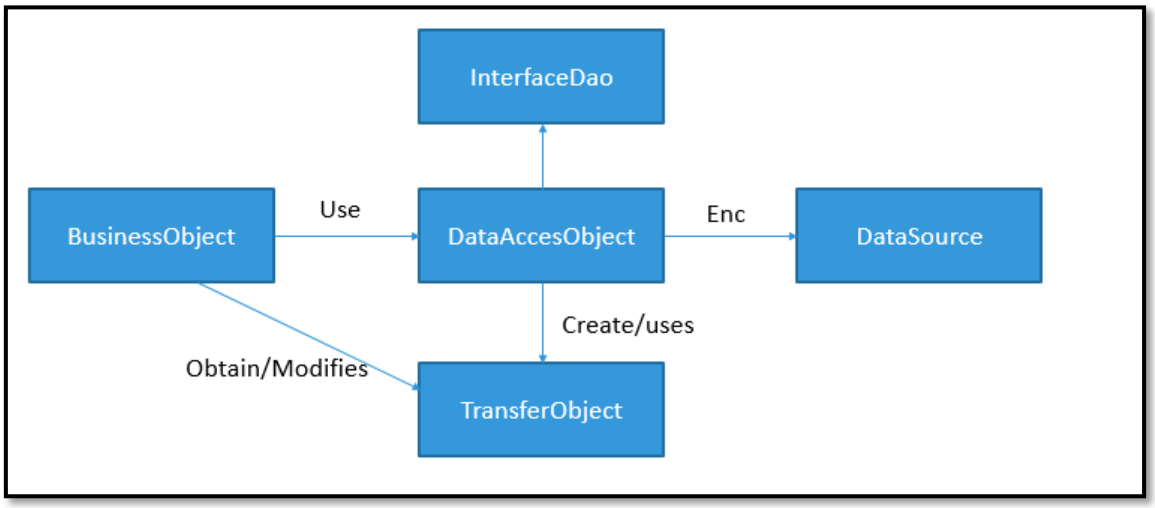



*Figura 29 Patrón DAO*

Una vez creado el modelamiento de la BD, se inicia la creación de los procedimientos para obtener los datos necesarios, para el mantenimiento del Ajustes de la Base Imponible:

| **Procedimientos Almacenados** | **Creación/Modificación** |
|---|---|
| proc_WEB_ReportingGIS_Select_AjustesDato | Creación |
| proc_WEB_ReportingGIS_Select_AjustesConcepto | Creación |
| proc_WEB_ReportingGIS_Modifica_AjustesBI | Creación |
| proc_WEB_ReportingGIS_Select_AjustesCategoria | Creación |
| proc_WEB_DiferidoAnual_VolcadoAjustes | Creación |

Se inicia de la definición de un WebServices para el manejo de los datos

En este apartado encontramos los componentes afectados de la lógica de negocio

| Development Component | Software Component | Track |
|---|---|---|
| srf/java/ibai/ajustebi | S007LNSR | T007GFLN |
| srf/j2ee/ejb/ibai/ajustebi | S007LNSR | T007GFLN |

### 3.4.3 LOGICA DE NEGOCIO COMPONENTES AJUSTES BI

Creación de tres clases:

DatoAjustesBi, CategoriaDTO y DatoAjustesBI. Cada una de estas clases tiene su método que es poder manejar y procesar los datos cuando llegan desde el Front-End como cuando se obtienen desde la capa de persistencia.

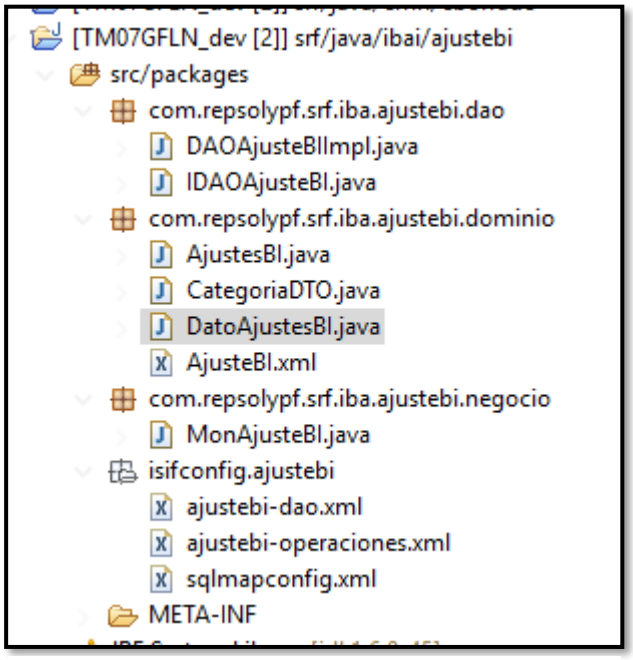



*Figura 30 Estructura componente AjustesBI*

En el paquete de dominio creamos el ajusteBI.xml que será el encargado de acceder a la base de datos a través Ibatis. Para ello se definen las llamadas a los procedures y sus correspondientes resultMaps como se muestra a continuación.

```
<resultMap id="datoAjusteBIResult" class="datoAjuste">
      <result property="intIdConceptoDato" columnIndex="1"/>
      <result property="chrConceptoDescripcion" columnIndex="2"/>
      <result property="chrDescripcion" columnIndex="3"/>
      <result property="bolPermanente" columnIndex="4"/>
      <result property="decAumento" columnIndex="5"/>
```

```
        <result property="decDisminucion" columnIndex="6"/>
          <result property="intIdConcepto" columnIndex="7"/>
        <result property="campoAdicional1" columnIndex="8"/>
        <result property="intIdCategoria" columnIndex="9"/>
        <result property="chrCategoria" columnIndex="10"/>
        <result property="chrTokenCategoria" columnIndex="11"/>
        <result property="chrAnoGeneracion" columnIndex="12"/>
        <result property="decRegularizacion" columnIndex="13"/>
        <result property="decSaldoInicialEjActual" columnIndex="14"/>
        <result property="decSaldoFinalEjActual" columnIndex="15"/>
</resultMap>

<procedure id="get" parameterClass="ajuste" resultMap="datoAjusteBIResult">
      {call
dbo.proc_WEB_ReportingGIS_Select_AjustesDato(#strUsuario:VARCHAR#,#intIdSocied
ad:INTEGER#,#intEjercicio:INTEGER#,#campoAdicional1:INTEGER#,#bolPermanente:VA
RCHAR#,#strIdioma:VARCHAR#)}
</procedure>
```

La implementación del DAO será la encargada de manejar los datos y controlar excepciones que podrían a ver surgido al acceder a la base de datos.

Luego se debe de modificar el xml de operaciones que es el encargado de conectar la llamada del EJB y redirigirla al método exacto que debe de ejecutar.

```
<bean id="IBAAJUSTESBI.GETCATEGORIA" parent="IBAAJUSTESBI">
   <property name="inputParameters">
      <list>

<value>INPUT,com.repsolypf.srf.iba.ajustebi.dominio.AjustesBI</value>
      </list>
   </property>
   <property name="methodName"><value>getTipos</value></property>
</bean>
```

En el ejb se definen los métodos para acceder a los mantenimientos de los ajustes de la siguiente manera

```
public CategoriaDTO[] invocarCategoria(AjustesBI entrada, String operacion) throws Exception{
    List lista = cliente.invoke("IBAAJUSTESBI."+ operacion.toUpperCase(), (Serializable)entrada);
    if(lista == null){ return new CategoriaDTO[0];}
    CategoriaDTO[] categoriaArray = new CategoriaDTO[lista.size()];
    lista.toArray(categoriaArray);
    return categoriaArray;
}
```

Igual que los métodos anteriores se crearán los métodos de mantenimiento para los ajustesBI, (GET, DELETE, UPDATE) y se realizaron sus respectivas implementaciones en la clase MONajustesBi.

### 3.4.4 WD IMPLEMENTACION

En este apartado se muestran los componentes afectados de la capa presentación.

| Development Component | Software Component | Track |
|---|---|---|
| repgf/srf/wd/ibai/ajustesbi | S007ISRF | T007GFIN |
| repgf/srf/wd/ibai/tabsindi | S007ISRF | T007GFIN |
| repgf/srf/wd/ibax/modelo | S007ISRF | T007GFIN |

**Ajustebi** = ajustes de la base imponible

**Tabsindi** = pestaña de sociedades individuales, es el componente superior que se encargara de conectar todos los componentes inferiores como es la ventana de ajustesBi, deducciones, HaidPID entre otras.


*Figura 31 Interfaz componente tabSindi*

**Modelo** = componente encargado de llevar todo el manejo de los datos como los mantenimientos. Este componente será usado por el componente principal tabsindi para manejar los datos de las distintas ventanas como puede ser una modificación, actualización y eliminación.

Una vez realizado la parte de la Lógica de negocio, se importar el modelo en cada uno de los DC.

En la componente Tabsindi se agrega la referencia hacia el otro componente ajusteBi de esta forma cada vez que se haga un clic en la pestaña ajustes BI del componente Tabsindi desencadenará una llamada al componente de ajusteBI y lo hará visible por pantalla.

En el componente modelo se agrega el WebServices de ajustes para acceder a los mantenimientos.

Model: El modelo es una estructura de datos que representan tanto los datos como las funcionalidades del banckEnd.

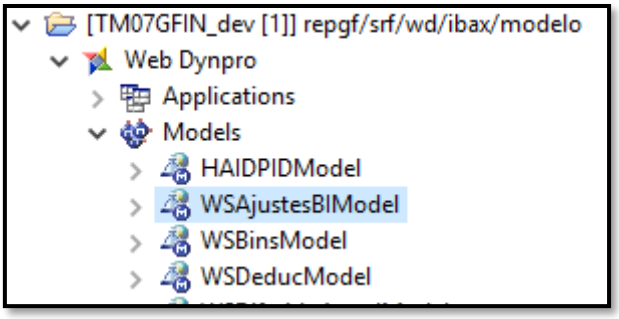



*Figura 32 Modelos importados en WD*

Esto acción crea los métodos necesarios para la llamada al WS como también construye las clases necesarias que soporta cada invocación del WS (respuesta del webService, excepciones, clases)

La siguiente imagen (Figura) muestra como se importa un WS el cual muestra las clases que tiene el WS como también las llamadas que se pueden realizar.

**Model Binding:** Es la vinculación entre el modelo y el controller

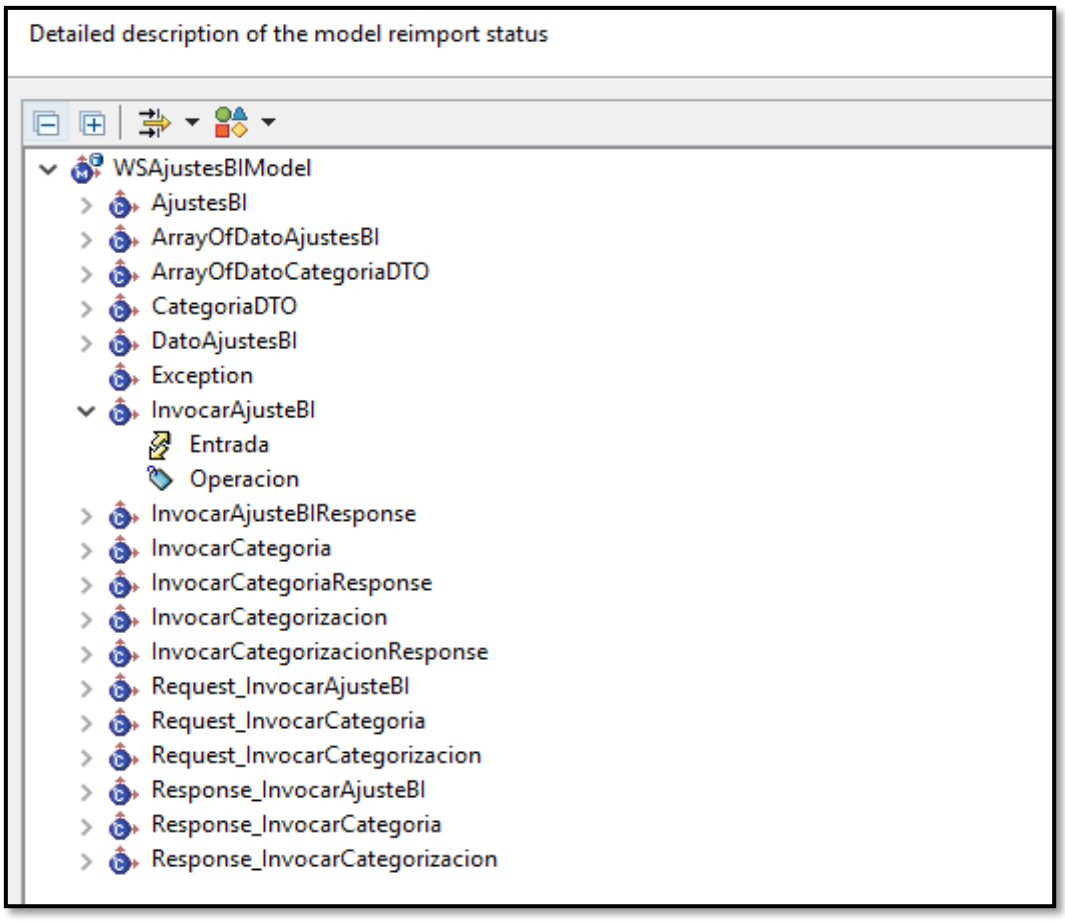



*Figura 33 Importación de Modelo en WD (Model Binding)*

Una vez terminado este proceso WD creara la estructura de Nodos que devuelve la operación que hemos importado del WS en el contexto del controller.

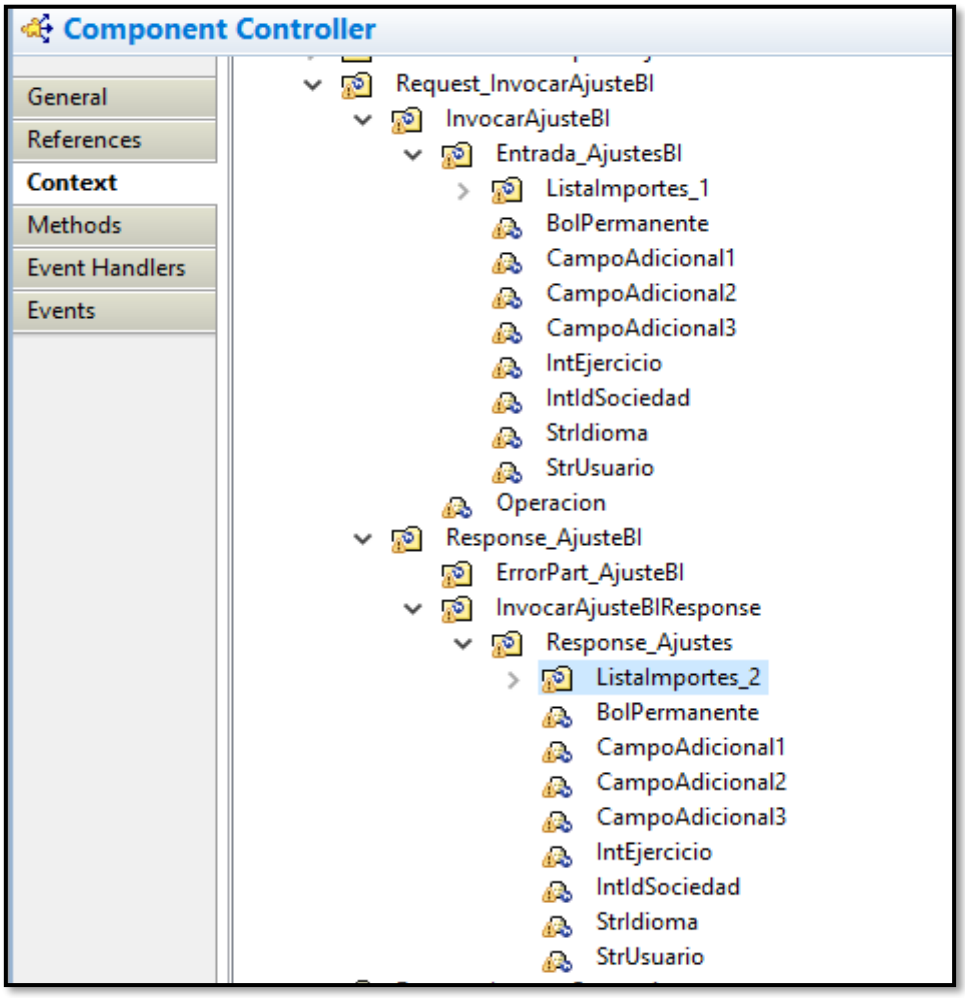



*Figura 34 Estructura de la respuesta del WS en nodos*

En el contexto del modelo se agregan nuevos nodos (**Context mapping**) al contexto (WD), estos nodos que contiene atributos se construyen para manejar los datos obtenidos por el WS y poder compartirlos entre componentes de nivel superior o inferior.

Un ejemplo de estas Nodos:

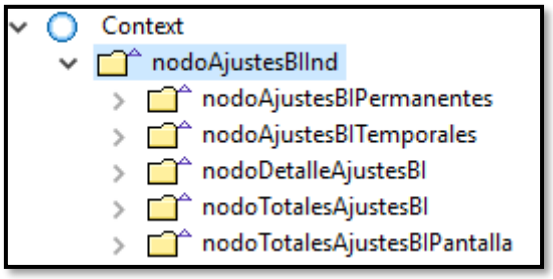



*Figura 35 Nodos del contexto*

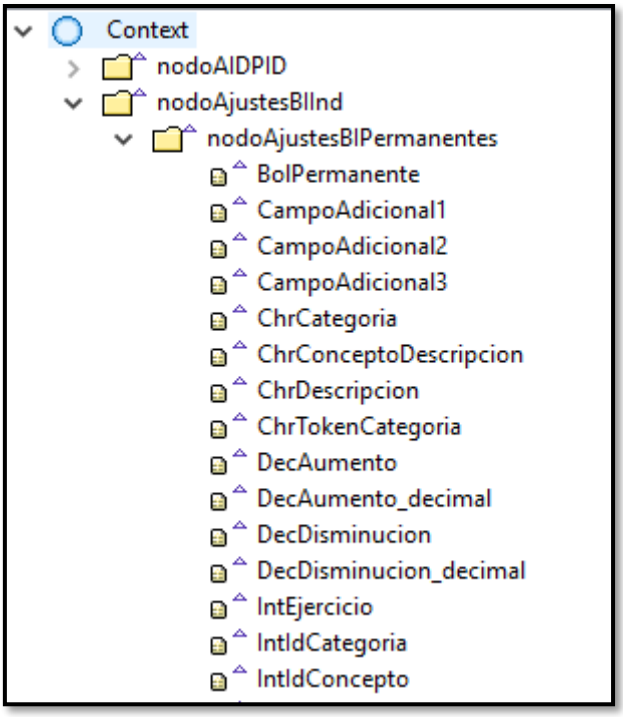



*Figura 36 Nodos del contexto - interfaz*

Luego con la ayuda de cada interfaz que es definida en cada componente podemos compartir la información entre ellos.

Las interfaces mantienen la misma estructura de nodos que se encuentra en el contexto del controller y con ello se puede acceder al context Mapping.

Cada vez que genere un evento por ejemplo se haga un clic en la pestaña de ajustesBI el componente que engloba todo el módulo Tabsindi obtendrá los datos a través del WD modelo que accede a los WS y devuelve los datos con la ayuda de las interfaces al Tabsindi y este a su vez los comparte con el componente WD ajustesBI. Para el guardado o modificación de datos se aplica el mismo mecanismo, pero con la ayuda de un evento que se canaliza al tabsindi y este se comunica con el modelo para hacer el mantenimiento de los datos.

En algunos casos algunos subcomponentes realizan invocaciones directamente a los WS ya que estas tareas no generan ningún recalculo entre las otras pestañas.

En el componente de Ajustes Bi con la ayuda de la paleta (UI Elements) se va diseñando la tabla, botones, textos. Es muy importante llevar un buen control de los IDs de cada elemento (tablas, labels, botones) para poder usarlos cuando se necesiten. (Figura tal)

Todos los subcomponentes definen una interfaz que contiene la estructura de sus Nodos no es obligatorio pero este caso lo es ya que se requiere para compartir los Nodos entre componentes, esta estructura puede ser accedida desde el componente principal tabsindi que será el encargado de copiar los datos al componente AjustesBI, con ello este poder hacer uso de contextMapping con la vista para poder mostrar los datos en

los UI elements. El proceso de asociar un atributo de un nodo con una view es llamada dataBinding.

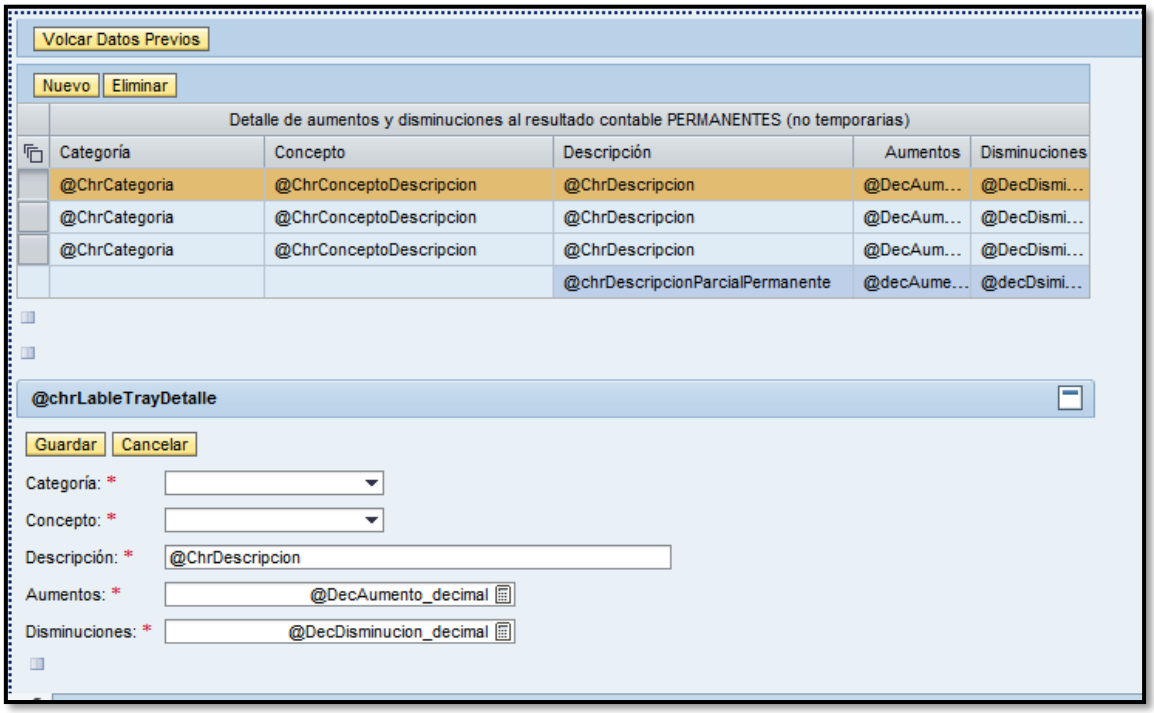



*Figura 37 UI elements*

Con la ayuda del DataLink podemos compartir estos nodos con la vista.

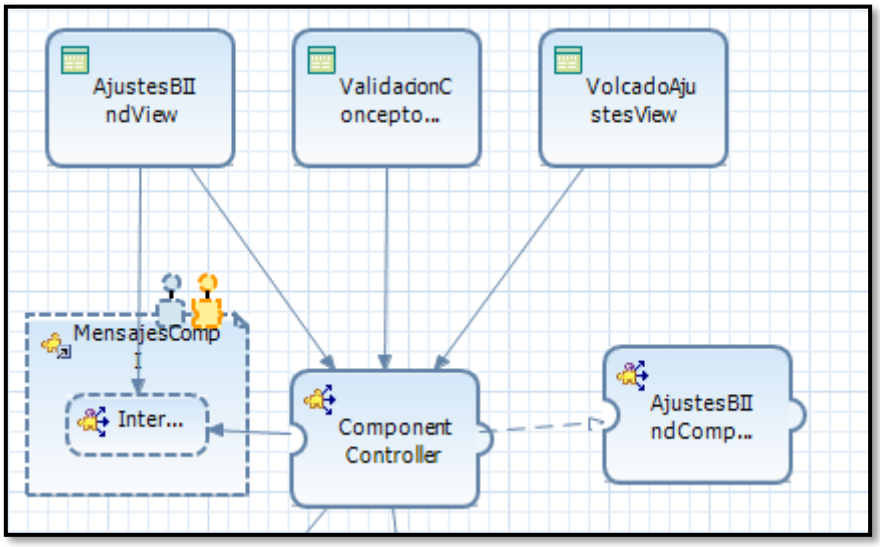



*Figura 38 Vista modelo de web Dynpro*

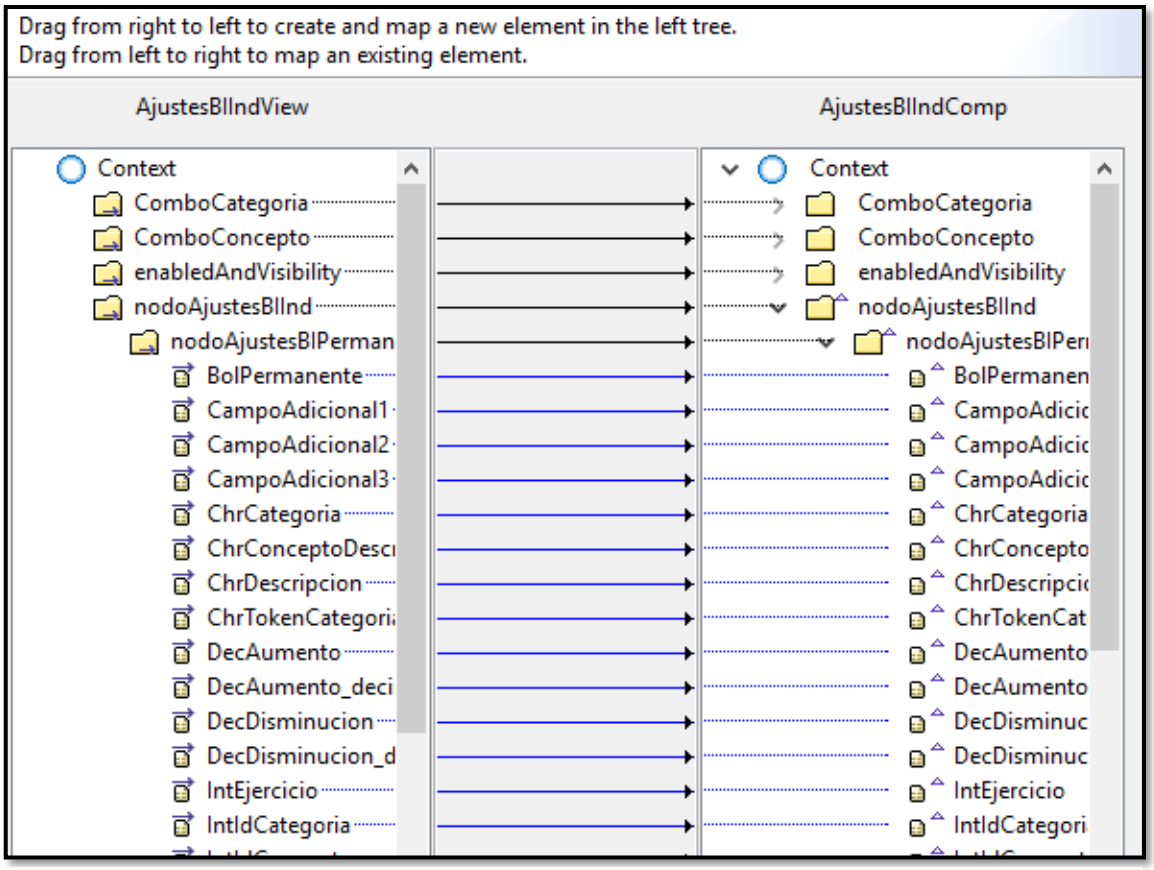



*Figura 39 Context Mapping*

Y desde la vista podemos por ejemplo a dirigir un determinado atributo a una celda

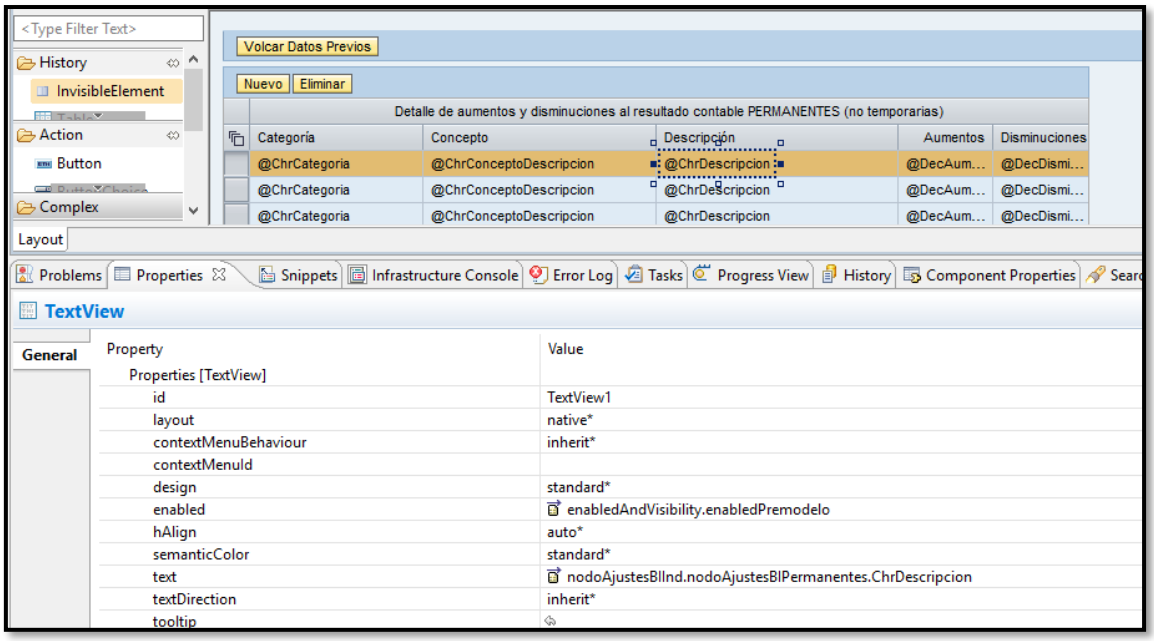



*Figura 40 Data Binding*

Lo mismo sucede con los botones que tienen atributos donde puedes hacer un BIND de información para cambiar un texto, hacerlo visible o no. Además de ello cada elemento tiene eventos predefinidos, ejemplo del botón Guardar onAction llamas a un método del controller para Guardar los datos de ajustes y este a su vez con un evento llama al método de guardar de Tabsindi que llama a al método guardar de ajustesBI del componente modelo.

A continuación, se detallan los requisitos:

Diseñar la tabla de ajustesBI Permanentes

| Detalle de aumentos y disminuciones PERMANENTES | | | | |
|---|---|---|---|---|
| Categorización | Concepto (desplegable) | Descripción | Aumentos | Disminuciones |
| IFRS | Ajuste permanente adaptación IFRS | Edificio Castellana | 200,00 | 80,00 |
| IFRS | Ajuste permanente adaptación IFRS | Edificio Arroyo | 150,00 | 50,00 |
| Ajustes Fiscales | Gastos no deducibles (multas, liberalidades, donativos) | Multa 1 | 100,00 | 0,00 |
| Ajustes Fiscales | Gastos no deducibles (multas, liberalidades, donativos) | liberalidades | 50,00 | 15,00 |
| Regularización ejercicios anteriores | Regularizaciones | Regularizaciones | 300,00 | 150,00 |
| Ajustes por Consolidación | Consolidación | Consolidación | 250,00 | 700,00 |
| | | | 1.050,00 | 995,00 |

*Figura 41 Tabla Ajustes Permanente*

Diseñar la tabla de ajustesBI Temporales

| Detalle de aumentos y disminuciones TEMPORALES | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Categoría (desplegable) | Año generación | Concepto (desplegable) | Descripción | Saldo Inicial Ejercicio Actual | Regularización | Aumentos | Disminuciones | Saldo Final Ejercicio Actual |
| Ajustes Fiscales | 2015 | Provisiones por insolvencias | Menor a 120 días | 500,00 | 100,00 | 200,00 | 0,00 | 800,00 |
| Ajustes Fiscales | 2015 | Limitación amortización 13 y 14 | Reversión en 15 años | 1.000,00 | 0,00 | 0,00 | 600,00 | 400,00 |
| Ajustes Fiscales | 2018 | Provisiones por insolvencias | Menor a 180 días | 0,00 | 0,00 | 800,00 | 0,00 | 800,00 |
| IFRS | 2018 | Ajuste temporal adaptación IFRS (activo) | Indemnizaciones no comunicadas | 0,00 | 0,00 | 500,00 | 300,00 | 200,00 |
| Ajustes Fiscales | 2018 | Limitación amortización 13 y 14 | Reversión en 10 años | 0,00 | 0,00 | 250,00 | 20,00 | 230,00 |
| IFRS | 2018 | Ajuste temporal adaptación IFRS (pasivo) | Amortizaciones | 0,00 | 0,00 | 1.000,00 | 0,00 | 1.000,00 |
| Ajustes por Consolidación | 2018 | Consolidación | Consolidación | 0,00 | 0,00 | 400,00 | 500,00 | -100,00 |
| | | | | 1.500,00 | 100,00 | 3.150,00 | 1.420,00 | 3.330,00 |

*Figura 42Tabla Ajustes Temporales*

## Modificar Detalle

- Año generación (Por defecto año del formulario) - obligatorio

  Como el componente principal tabsindi, maneja los datos iniciales del usuario podemos obtener el año desde la búsqueda principal de Carga Sociedad Individual

- Añadir Categoría (obligatorio)

  El campo debe estar precargado con los datos de base de datos. Al seleccionar un elemento del combo cargará el combo de conceptos. Mientras no haya seleccionada una categoría, el combo de Conceptos estará vacío.

  Se hará uso del componente estándar WD de combos para la carga de los combos de categoría y conceptos. En un inicio cuando este cargada la vista de ajustesBI mediante el ciclo de vida wdInit() se llamará al método getCategorias del controlador para obtener la lista de categorias (id y la descripción), gracias al contextMapping podremos tener los atributos ligados con el combo de la

vista. Una vez el usuario seleccione alguno de los categorias se obtendrán la lista de conceptos ligada a esta categoría y se cargara al combo de conceptos.

- Regularización: (Deshabilitado si el año inicial es distinto de año de la búsqueda.)

  Al seleccionar un registro de los datos se cargan en la pantalla detalle donde se comprueba que si el año de generación del detalle es igual al año de la vista de la búsqueda se mantendrá deshabilitado en caso contrario se habilitara.

- Saldo Inicial Ejercicio Actual (Sólo lectura e inicializar a cero)

  Se mantendrá deshabilitado por medio del atributo de visibilidad solo podrá ser modificado cuando se haga un volcado de datos por medio de un interfaz (proceso de carga de datos por medio de xml)

- Saldo final Ejercicio Actual (Sólo lectura y calculado): Saldo Inicial" + "Aumentos" - "Disminuciones

  Desde la ventana del detalle se calculará el saldo Final o cuando se de guardar en el tipo de ajustes.

- Aumentos y Disminuciones (Obligatorios)

  Campos obligatorios, solo admiten números y si el usuario le da a la tecla enter debe calcular el saldo final del ejercicio actual.

  Para ello se define un evento onEnter que sumara las calculara las casillas aumentos – disminuciones y el resultado se mostrara saldo Final Ejercicio Actual.

Crear un Ajuste Permanente o Temporal

El usuario al rellenar todos los parámetros obligatorios guardara el ajuste el cual pasara por una comprobación previamente para validar los datos. Una vez pasado a estas validaciones se envía un evento al componente padre TabSindi el cual lee el ajuste y lo envía al modelo el cual lo persiste en BD y recalcula los totales y los devuelve a la interfaz de ajustesBI.

Eliminar o Modificar Ajustes:

El usuario seleccionara uno de los elementos de la tabla el cual carga los datos los datos en el formulario, una vez el usuario modifique los datos este pasara

por una validación para comprobar si los datos han sido modificados o no y lanzar un evento de guardado al componente padre.
En el caso de una eliminación se lanza un evento al componente padre y este termina en la base de datos donde por medio del id se elimina los ajustes seleccionados.

Los botones de la tabla Nuevo y Eliminar:

Nuevo solo podrá ser usado si la sociedad no está en fase de consolidado en caso contrario no se podrá hacer uso del botón.

Botón eliminar solo podrá ser activo cada vez que se selecciona un registro de la tabla.

Al guardar Detalle:

- Se debe tener en cuenta el manejo de los totales y el correcto manejo de los IDs tanto para el campo concepto como de categorización.
- Si el usuario cancela la acción de modificar esto no debe afectar ni hacer ningún campo.
- Cada guardado de un nuevo ajuste se guardará directamente en la BD y actualizará las vistas correspondientes

**Volcar Datos Previos**

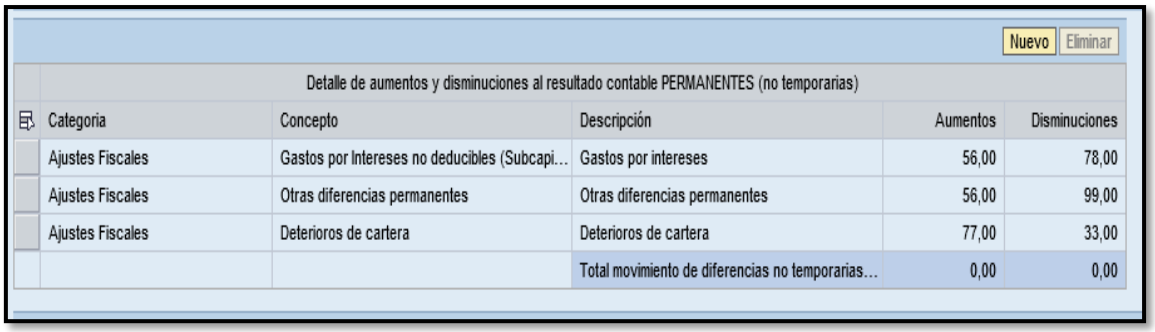
Se creará un nuevo botón “Volcar datos previos” que afectará únicamente a la tabla de ajustes temporales.

El botón emite una llamada a la lógica de negocio donde se obtendrán los periodos de donde tendrá que obtener los datos de periodos donde tiene que volcar los datos.

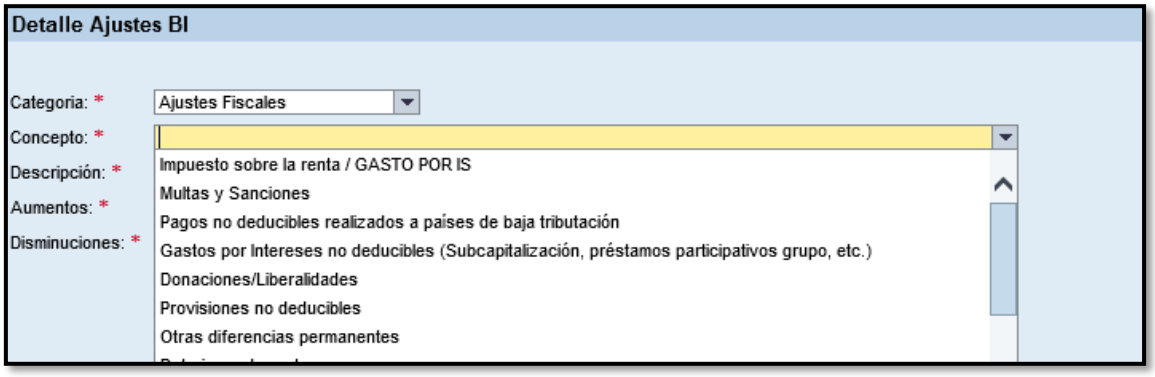
El siguiente botón existe tanto en la pestaña deducciones, ajustes y BINS.

Cada vez que se intente volcar los datos previos se deberá mostrar al usuario un popUp

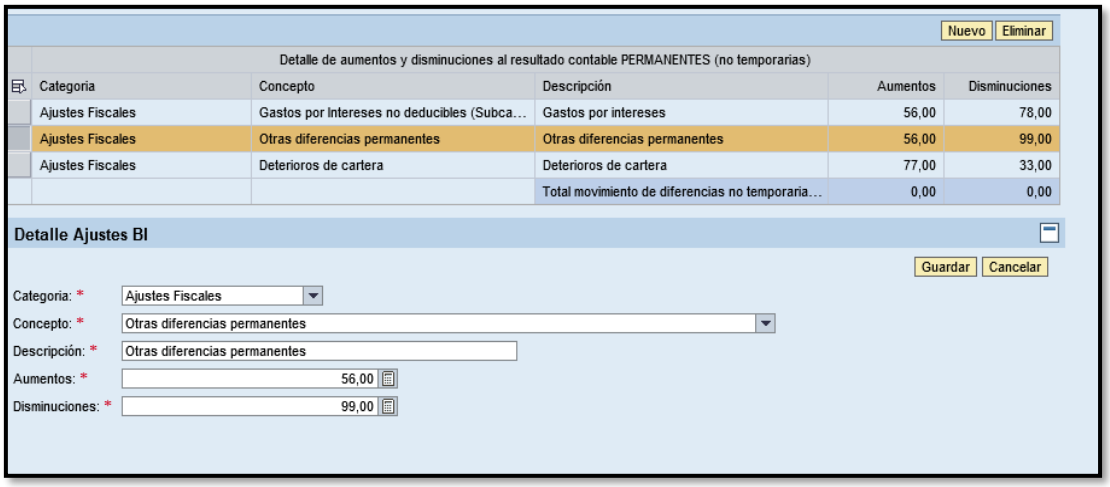
- Diseño de tabla ajustes Permanentes:



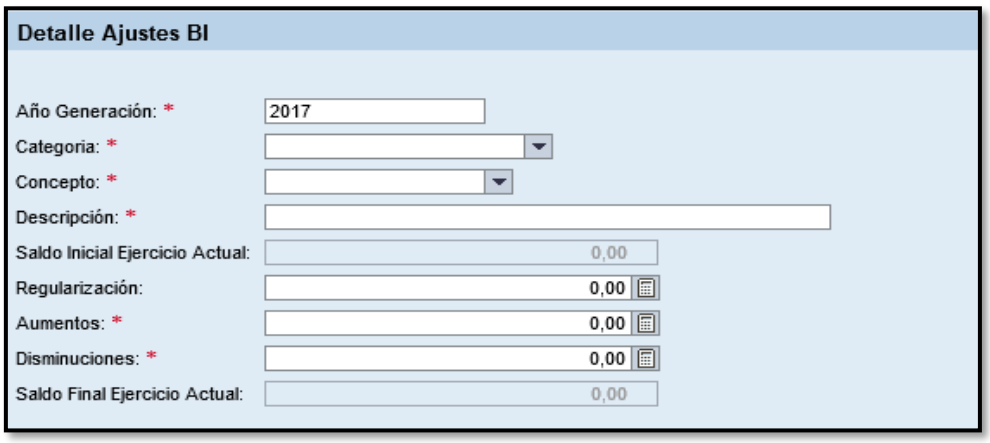
- Modificar y Crear Ajustes:
  - Al seleccionar un elemento del combo categoria cargará el combo de conceptos. Mientras no haya seleccionada una categoría, el combo de Conceptos estará vacío.




Modificar detalle:



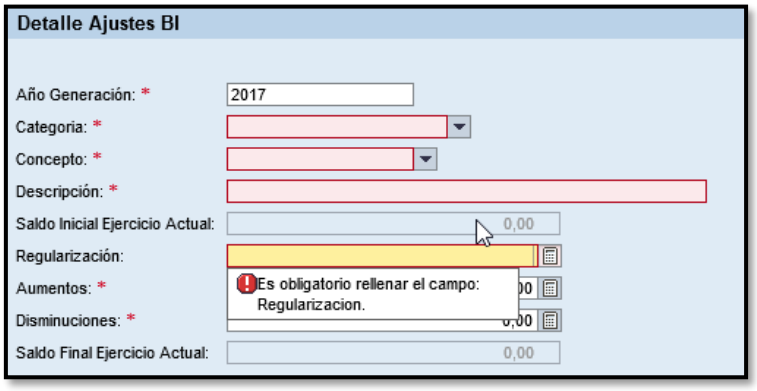

– Formulario de ajuste temporales:



Aparte de añadir el nuevo atributo en el nodo, se deberá modificar el método cargarPermanentesAjustesBI del controlador ModeloComp para recoger los nuevos atributos.

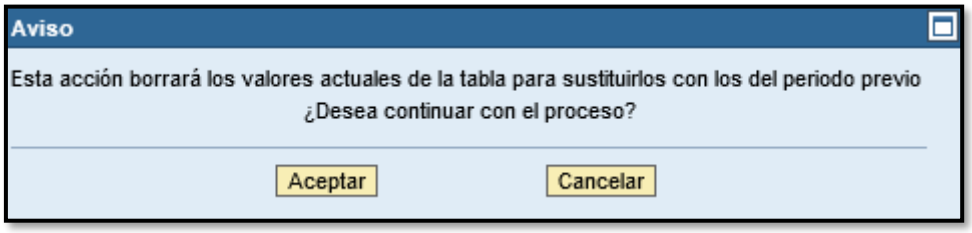
- Validaciones de los formularios:

Por medio del botón guardar se comprueban desde la vista todos los campos que sean correctos antes de acceder a la lógica de negocio.



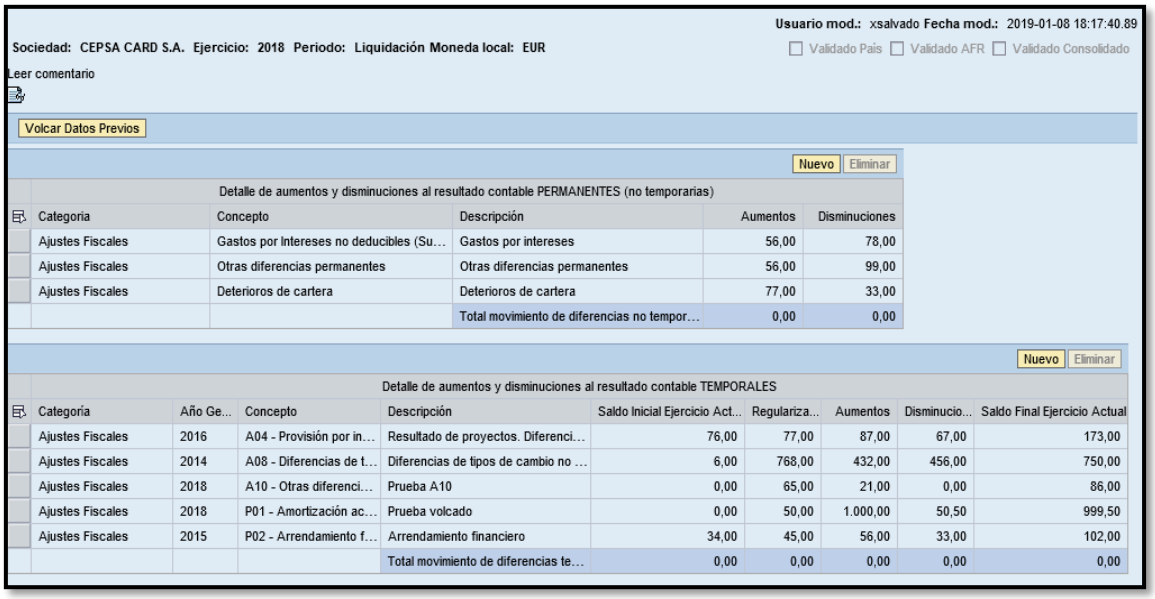

- Botón Volcar Datos Previos a ajustes Temporales: Al presionar el botón de volcados BI. Se creará un método en la view para llamar al método del controlador. El método del controlador deberá invocar a la operación VolcarDatosAjustesTemporales del WS.


Mensaje de advertencia al usuario de la acción que se va a tomar:




- Para todos los campos añadidos anteriormente se deberá modificar los ficheros XXX.wdview_en.xlf para soportar las traducciones en inglés.

### 3.4.5 DISEÑO FINAL




## 3.5 ANALISIS DE MIGRACION DE LA APLICACION

Actualmente SAP ha dado un año más de soporte a la tecnología de Web Dynpro que ha sido reemplazada por SAP HANA y SAP UI5. Es por ello que a lo largo de este año y del año anterior se ha estado realizando una evaluación tanto de arquitectura como de costes que conlleva evaluar distintas soluciones para la migración de la aplicación de SIGEGI. Algunas de las tecnologías que sea han estado evaluando tanto para el Front-End como para el Back-End son las siguientes:

### Front-End

- Angular 7: Framework desarrollado en JavaScript o TypeScript. Como punto positivo es que el personal de Indra tiene conocimientos y experiencia con el framework por lo que no sería necesaria para los programadores empezar desde cero, además de ello se podría aprovechar usar Bootstrap y Material Design que facilitaran tanto la transición entre dispositivos de visualización como pueden ser tables, móviles etc. También se podría aprovechar implementar una lógica interna entre los formularios que permitiría cálculos o comprobaciones sencillas sin necesidad de ir al servidor por lo que se ganaría velocidad y la liberación de recursos al servidor. Por otro lado, usar el DataBinding que permite compartir el valor de un concepto entre las diversas pestañas que compongan el formulario sin necesidad de ir al servidor cada vez que cambiemos de pestaña.

- React: Es una librería de JavaScript que es mantenida por Facebook, está orientada a la visualización no es un framework como tal, pero en conjunto con

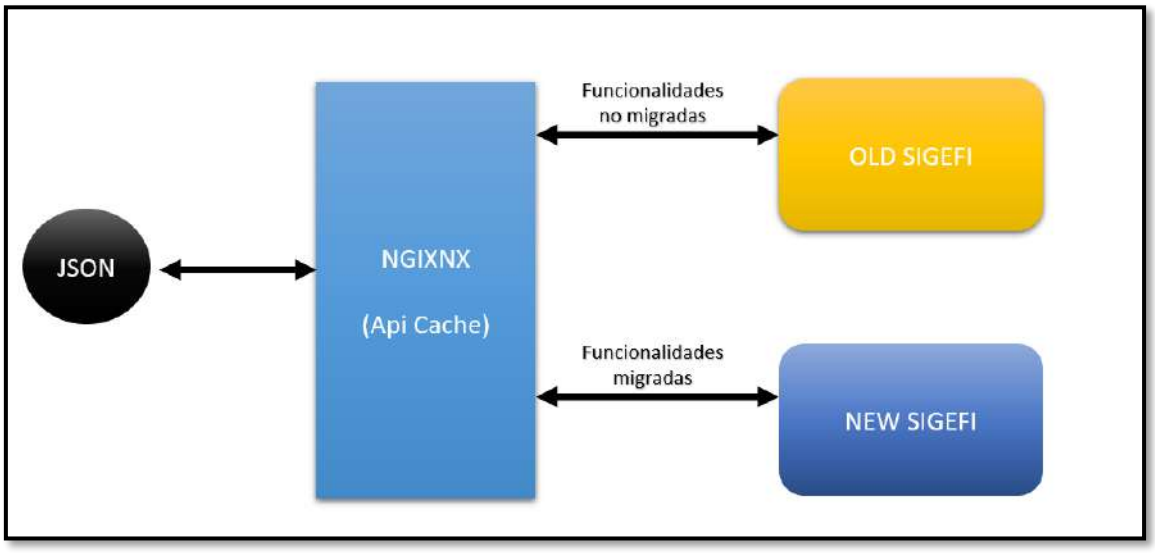
otras tecnologías formas un framework REACT + REDUX + RECT ROUTES. Un problema que se encuentra es que muy pocas personas en Indra saben acerca de esta tecnología por lo que no hay personal con experiencia. React es mucho más rápido en la visualización de contenidos que Angular gracias a su mecanismo Virtual DOM.

- SAP UI: Es un framework que es mantenido por SAP y que es basado en JavaScript, jQuery entre otros está orientado a aplicación empresariales. El objetivo que tiene es mejorar la experiencia del usuario.

  Como un extra si se desearía en un momento centrarse más en el comportamiento a dispositivos móviles, se podría aplicar el framework de IONIC sobre el código ya generado con Angular para crear el código nativo al dispositivo, por otro lado, para REACT podríamos usar el framework de React Native o IONIC sobre el código ya generado.

## Integración con el Front-End

Al ser Sigefi una aplicación que tiene varios años de desarrollo se propone que mientras no se acabe la sustitución de una versión por la otra, se mantenga la versión actual como la nueva, dependiendo de a que funcionalidad quiera acceder el usuario el servidor redirigirá a los módulos que aún no estén migrados o a los módulos desarrollados en las nuevas tecnologías para reducir el impacto a los usuarios y mantener la aplicación funcionando.




Como mecanismo de transporte de información de usuario JSON, que es el estándar en el transporte de la información por red.

Para las comunicaciones se dividirá las comunicaciones en dos tipos:

- Contenido estático; imágenes, código JavaScript estático, etc.

- Contenido dinámico, todo lo que tenga que ver con cálculos, etc.

Para distinguir los dos tipos de peticiones, el punto de entrada será NGINX (Servidor web proxy / inverso) y que a su vez nos podría servir como api cache.

Además, para servir el contenido dinámico se están evaluando las siguientes tecnologías:

- **JSON + reactive Streams**: Desde la versión Spring Framework 5 y desde versiones recientes de SpringBoot, Spring permite seguir el patrón de arquitectura Reactive, es decir, bajo demanda ante un evento, se ejecuta el código correspondiente a un evento cuando este evento se produce, evitando el bloqueo de recursos cuando no hay necesidad. Se seguirá el patrón REST de uso semántico de los puntos de acceso a lógica de Negocio y Datos y JSON como formato para el transporte de la información.
- **JSON + SpringMVC**: Para la integración entre las partes Front y Back se usará JSON, como formato de la información transportada, y SpringMVC como gestor de recepción y envío de datos (conversión de JSON a objetos, gestión de cabeceras, forwarding...). Se seguirá el patrón REST de uso semántico de los puntos de acceso a lógica de Negocio y Datos. El uso de JSON y REST es el estándar de facto en el mercado.

## Lógica de Negocio:

Las unidades lógicas como recomendación se basarían en SpringBoot como plataforma para conseguir la máxima escalabilidad, facilidad de interconexión y reaprovechamiento de código ya existe.

Se pueden realizar diversas combinaciones de integración, dependiendo de si integramos la lógica de negocio con el acceso a base de datos en una unidad lógica de ejecución, o de si integramos la capa que da servicio al Front con la lógica de negocio y con el acceso a base de datos en una unidad lógica de ejecución, o si finalmente no integramos ninguna capa y cada una de ellas se ejecuta en unidades lógicas de ejecución distintas.

### Acceso a Base de Datos:

Se mantendría SQL con el framerwork de myBatis que permitiría una fácil integración con Spring. Esta integración incluye participar en las transacciones gestionadas por Spring, crear los Mapas y las sesiones de comunicación con la BBDD entre otros. Se debe tener en cuenta que MyBatis, a diferencia de JPA no es un ORM (Modelo Objeto-Relación), con lo que se ahorra la sincronización entre las tablas en la BBDD.

# 4 BLOQUE DE FINALIZACION

## 4.1 CONCLUSIONES Y TRABAJO FUTURO

### 4.1.1 CONCLUSIONES PERSONALES

Formar parte del proyecto me ha brindado mucha experiencia, ya que he podido ver como se realiza un desarrollo mediano para una empresa, y cuales son todos los pasos que se siguen hasta la finalización del proyecto, iniciando desde la venta, toma de requisitos, diseños, pruebas, puesta en marcha de la aplicación y el mantenimiento.

El aprendizaje de la tecnología WD y del entorno en general no ha sido fácil de asimilar, me ha tomado tiempo acostumbrarme, y a poder desarrollar de una forma más fluida con esta tecnología nueva, pero día a día he podido encontrar muchas similitudes con otras tecnologías que me han ayudado a desenvolverme mejor en el desarrollo del proyecto. A medida que se iba desarrollando el proyecto, las estimaciones fueron aumentando, no solo por error del cliente al no saber con exactitud que quería, sino también por temas de comunicación entre los equipos de técnicos y funcionales, o en algunos casos por errores en programación de validaciones. Con la ayuda de las metodologías agiles, estos problemas se solucionan de manera rápida ya que se mantiene comunicación con el cliente y el equipo funcional en todo momento desde el inicio del desarrollo y en la evolución del mismo.

En definitiva, la experiencia ha sido muy buena, he aplicado conceptos aprendidos en la universidad, como también he aprendido nuevas tecnologías como es WD y Java EE, además de ello me he podido percatar lo mucho que es importante la comunicación entre todo el equipo. A lo largo del desarrollo han ocurrido conflictos entre equipos, que han ayudado al equipo en conjunto a detectar y minimizar distintos tipos de errores para que en futuros desarrollos no sucedan nuevamente, como equipo esto nos favorece ya que aumenta la calidad del producto que se le brinda a clientes actuales y potenciales.

### 4.1.2 TRABAJO FUTURO

Como trabajo futuro, el proyecto actualmente tiene una carencia que es la parte del Front-End la tecnología actualmente usada Web Dynpro está llegando a su fin, esto quiere decir que el próximo año ya se dejara de prestar soporte de parte de SAP, por lo que ya existe un plan de acción donde se están evaluando tecnologías Front-End y aprovechar el momento para poder hacer una actualización del Back-End a versiones superiores. Con esto se busca no solo contar con una tecnología que se mantenga mejor en el tiempo, si no también brindarles a los usuarios una mejor presentación de la aplicación que sea más agradable y más usable.

Entre las tecnologías que se están evaluando para el Front-End están las siguientes:

- Angular
- React (REACT + REDUX + RECT ROUTES)
- Sap UI

Dentro del Back-End se está buscando mantener lo actual, pero migrando a nuevas versiones:

- Spring
- Ibatis
- Java

La migración de la aplicación se ira haciendo de forma paulatina, siendo migrada a trozos, para minimizar el impacto a los usuarios, esto quiere decir que se mantendría la versión actual en WD y los nuevos desarrollos se irían desarrollando con las nuevas tecnologías seleccionadas, y por último se iría moviendo módulos de la tecnología antigua a la nueva.

# 5 BIBLIOGRAFIA


1. Web Dynpro

https://help.sap.com/saphelp_nw70/helpdata/en/77/3545415ea6f523e10000000a155106/frameset.htm

2. Modelo-Vista-Controlador
https://developer.mozilla.org/enUS/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

3. Java EE https://javaee.github.io/tutorial/toc.html

4. Scrum https://www.agilealliance.org/glossary/scrum/

5. kanban https://www.agilealliance.org/glossary/kanban/

6. Jira https://confluence.atlassian.com/jirasoftwarecloud/jira-software-overview779293724.html

7. Netweaver eclipse http://www.teknodatips.com.ar/sap-netweaver/285-abap-en-eclipse-entorno-desarrolloadt.html

8. https://openui5.org/